195 PTAS.
(IVA Incluido)

MI COMPUTER

115

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

P.V.P. Canarias, Ceuta y Melilla 185 Ptas.



Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen X-Fascículo 115

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones
Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 120 fascículos de aparición semanal, encuadernables en diez volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-183-6 (tomo 10)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 018604
Impreso en España-Printed in Spain-Abril 1986

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (120 fascículos más las tapas, guardas y transferibles para la confección de los 10 volúmenes) son las siguientes:

a) Un pago único anticipado de 27 105 ptas. o bien 10 pagos trimestrales anticipados y consecutivos de 2 711 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.

Editorial Delta, S.A.



Intuición e imaginación

Se asegura que la aparición del Commodore Amiga constituirá un acontecimiento debido a su innovador diseño y excepcional rendimiento



Marcus Wilson-Smith

Un salto evolutivo
El Commodore Amiga representa un notabilísimo desarrollo en la evolución del microordenador. El uso intensivo de chips construidos a medida, aplicando tecnología *blitter*, mejora enormemente las capacidades de gráficos y sonido de la máquina, mientras la CPU queda libre para manejar otros procesos

El Amiga fue desarrollado originalmente por la firma norteamericana Amiga Corporation y posteriormente sus derechos fueron adquiridos por Commodore. Si bien la máquina ha aparecido en Estados Unidos bajo el patrocinio de Commodore, ninguno de sus distintivos menciona el nombre de la compañía. Debido a que el ordenador está dirigido al mercado de gestión o al usuario serio, y a que en dicho país Commodore está considerada como un pequeño fabricante de máquinas de juegos (en gran parte a consecuencia del éxito del Commodore 64), la empresa decidió comercializar la máquina con una imagen completamente nueva.

Alojado en una única carcasa con una unidad de disco incorporada, el Commodore Amiga tiene un aspecto muy de gestión. El teclado, que está separado del ordenador y conectado mediante un generoso trozo de cable, tiene patas en la parte posterior, que se pueden levantar para elevar en ángulo las teclas y obtener mayor comodidad al teclear. Si bien el tacto del teclado no es el mejor de los existentes, ciertamente es adecuado para la mayoría de las aplicaciones. Abajo y a la derecha de las teclas de máquina de escribir hay un grupo de teclas de cursor que se puede utilizar para reproducir las funciones del ratón de mano.

El Amiga viene con una unidad de disco de 3½ pulgadas de doble cara, en la que los fabricantes han conseguido insertar hasta 880 Kbytes de datos. Esto permite al Amiga acceder a más datos en una sola unidad que muchos ordenadores con unidades gemelas.

El panel frontal a la izquierda de la unidad revela, al quitarlo, una ranura para ampliación. Dotado con 256 Kbytes de memoria como estándar, el Amiga puede acomodar un módulo de memoria adicional que simplemente se introduce en el panel frontal, con lo que la máquina alcanza los 512 Kbytes. Para ampliar el Amiga hacia la configuración máxima de ocho Mbytes, en el lado derecho de la carcasa del ordenador se encuentra una segunda ranura a través de la cual se puede conectar en interface la memoria adicional. En este lado de la máquina también se proporciona un par de conectores D de nueve patillas para palancas de mando o, lo más importante, para el controlador de ratón.

La parte posterior aloja las conexiones para interface de periféricos. Desde la izquierda, éstas son la puerta para teclado, la puerta para impresora Centronics y una interface para una segunda unidad de disco. Asimismo, el Amiga posee una conexión RS-232C para modem y otros periféricos en serie y un par de conectores de tipo *phono*. Éstos se pueden enchufar directamente en la pantalla especializada que se proporciona para el Amiga o, lo que es más conveniente, en un sistema de *hi-fi* en el cual se podrá apreciar cabalmente la calidad del sonido del ordenador.

Las tres puertas restantes están reservadas para funciones de video, incluyendo un conector de 23 vías para monitores RGB y un conector para video compuesto. También hay disponible un conector *video in* que permite que el Amiga entre imágenes de video desde un VCR y visualice encima de esas imágenes otras generadas por ordenador, algo así como el sistema Pioneer PX-7 (véase p. 2069). Es probable que entre los futuros accesorios de hardware para el Amiga se incluya un *frame-grabber* (manipulador de ventanas) que digitaliza un fotograma de una fuente de video, que después se puede reflejar, rotar o tratar de otro modo bajo control de software.

En el centro del Amiga está el procesador 68000 de Motorola (como en el Apple Macintosh y el Atari 520ST), pero lo que en realidad le proporciona al Amiga su sobresaliente rendimiento es la influencia que ejercen, entre sí y sobre el 68000, tres chips hechos a medida. Denominados Agnus, Portia y Daphne, los chips pueden controlar visualización de video, sprites y E/S de disco de forma más o menos autónoma, dejando libre al 68000 para que lleve a cabo las aplicaciones de proceso a toda velo-



Todo en las teclas
El teclado que viene con el Amiga es uno de los mejores que hay actualmente en el mercado. Siguiendo las tendencias modernas, incluye un teclado numérico, 10 teclas de función programables y otras numerosas teclas que se pueden utilizar en funciones de control

La firma Metacomco

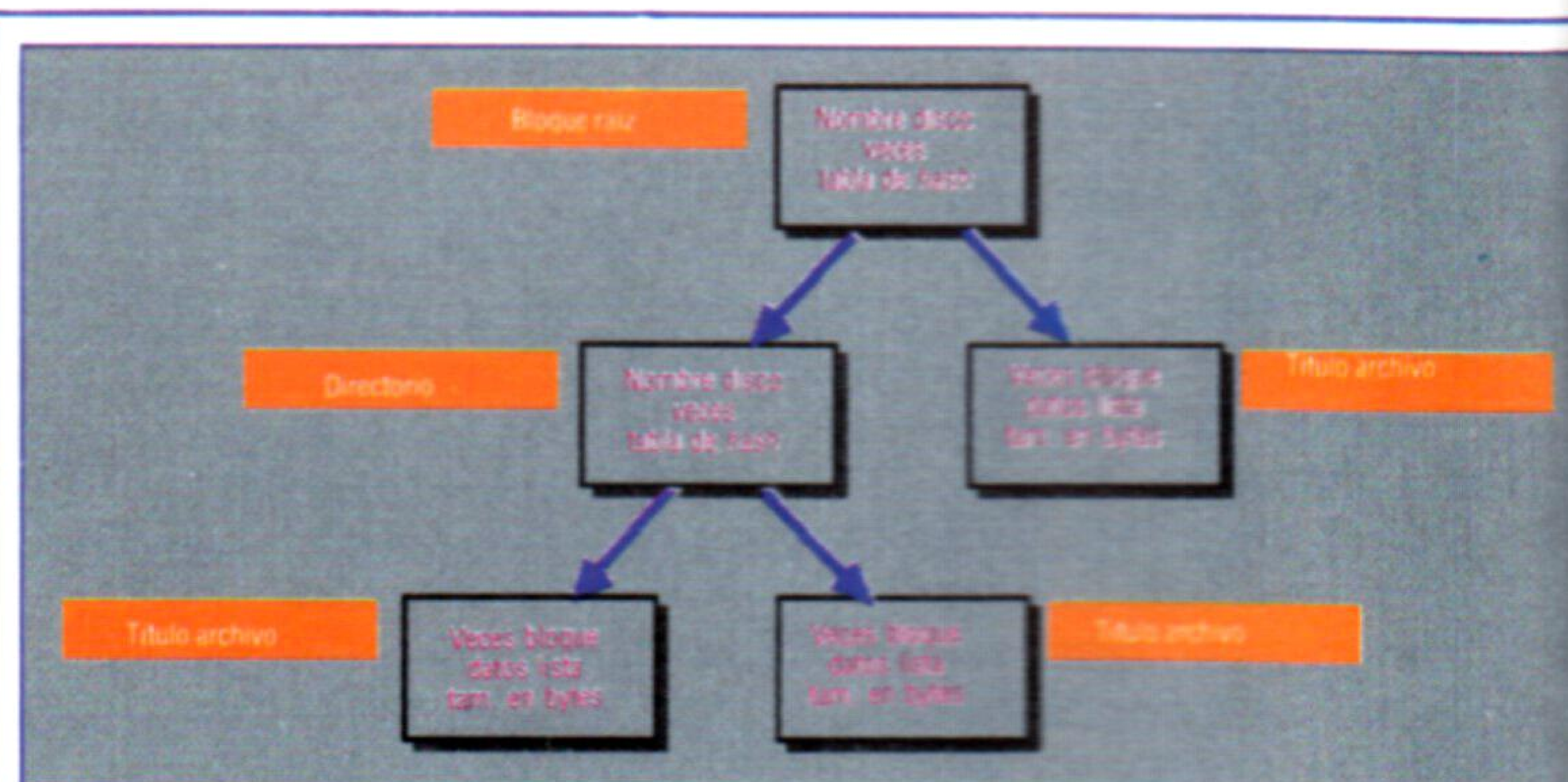
La casa de software Metacomco, con sede en Bristol, se creó en 1981 para desarrollar software de sistemas para ordenadores de 16 y 32 bits. Su Personal BASIC, desarrollado para Digital Research, se ha convertido en un estándar en ordenadores que ejecutan CP/M-86. Más recientemente, Metacomco ha creado software 68000 y producido versiones de PASCAL, LISP y C para el Sinclair QL y el Atari 520ST, así como ensambladores 68000 y paquetes para desarrollo de software. Metacomco fue contactada para escribir el AmigaDOS para el bisoño Amiga cuando otra casa de sistemas, contratada inicialmente para desarrollar un OS, falló en entregar el producto. El AmigaDOS se basa en un sistema de red multitareas conocido como Tripos, creado hacía algunos años en la Universidad de Cambridge. Al doctor Tim King, jefe del departamento de I&D de Metacomco, se le

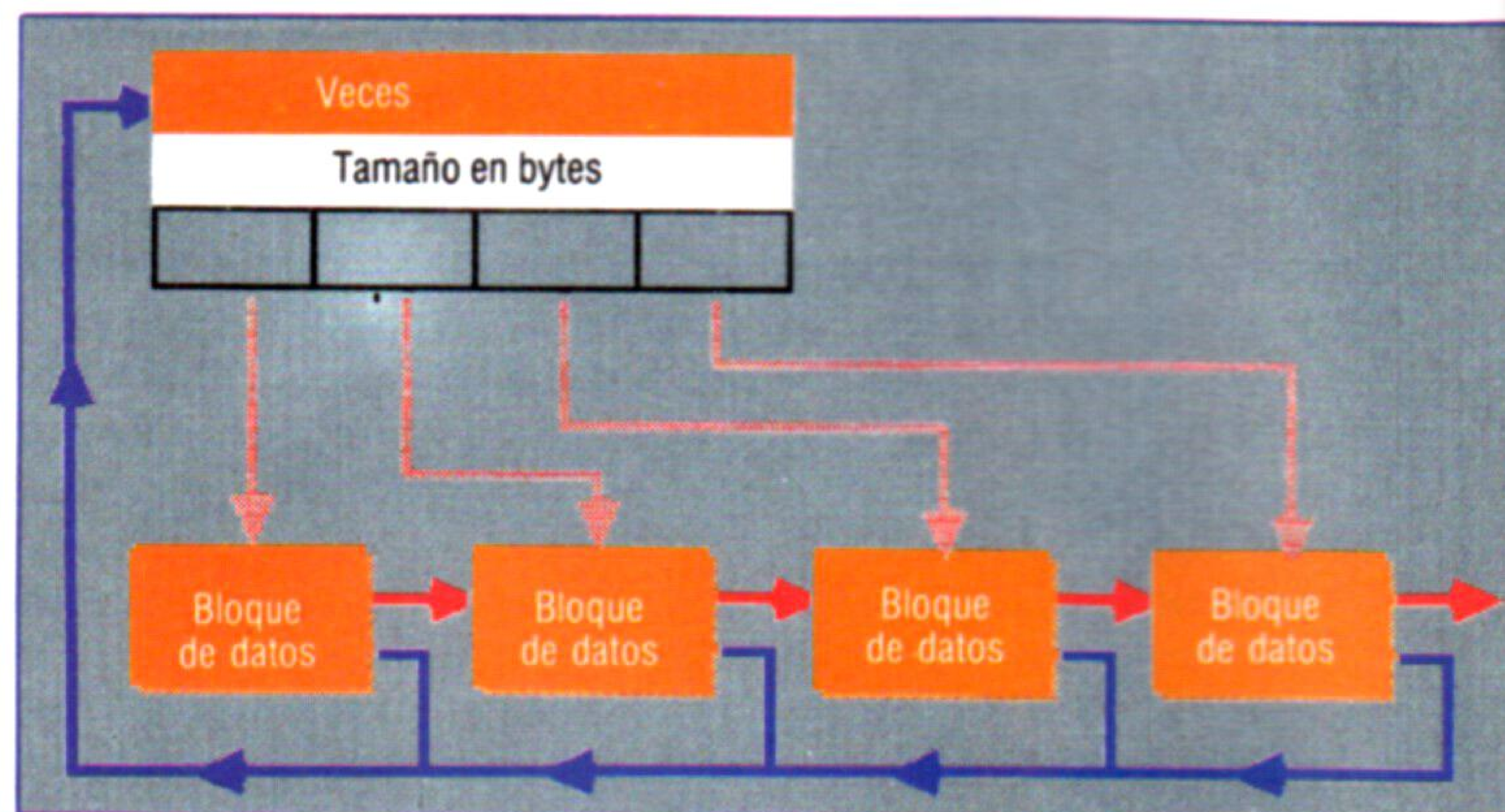
cidad. Agnus, por ejemplo, posee su propio coprocesador y «manipulador de imagen de bits» (*blitter*) que le permite mover un millón de pixels por segundo. La lógica para dibujo de líneas y relleno de formas también forma parte del sistema de circuitos de Agnus. Todo esto significa que el Amiga posee la capacidad de mover, alterar y rellenar formas tan rápidamente que crea la ilusión de hacerlo de forma instantánea. Además, el blitter se utiliza para transferir datos de disco entre los *buffers* de disco y la memoria.

Hay disponibles dos tipos de sprites. Los primeros, conocidos como *Vsprites*, se controlan desde hardware, permitiendo su movimiento alrededor de la pantalla a gran velocidad. El segundo tipo de sprites se controla desde software utilizando el blitter. Estos "objetos de blitter" (*bobs*) permiten formas y coloreados más complejos de los que se pueden conseguir con los Vsprites. La suma de todas estas características hace que el Amiga pueda producir gráficos a una velocidad extraordinaria y de una calidad sin precedentes, que antes sólo se podían obtener en máquinas recreativas muy especiales.

Las capacidades de sonido también son notables, de una calidad equiparable a la de muchos sintetizadores comerciales. El Amiga puede reproducir digitalmente sonidos reales muestreados que se pueden manipular en estéreo en cualquier altura dada. También se incluye como estándar la síntesis de voz, que puede hablar con voces masculinas o femeninas, añadir inflexiones y procesar texto escrito. Dos de las aplicaciones que siempre saltan a la imaginación son procesadores de textos que posean la facultad de leerle texto al usuario o de leerle los mensajes de su buzón de correo electrónico. Pues bien, el BASIC que se proporciona con el sistema (AmigaBASIC) soporta todas estas facilidades y permite producir gráficos excelentes a partir de unas pocas líneas de programa. La síntesis de voz la soportan instrucciones de BASIC que traducen un texto a una serie de fonemas, que después son "hablados".

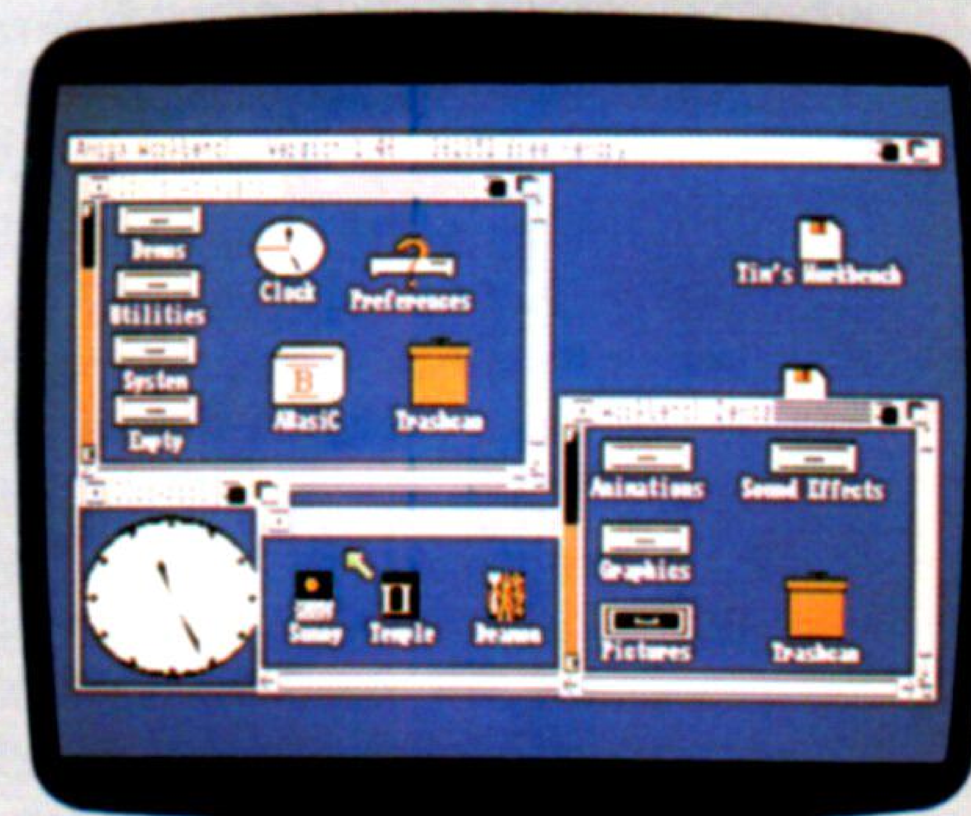
El Amiga presenta al usuario un entorno tipo WIMP amable, llamado Workbench, que sigue el estilo del Macintosh y las máquinas basadas en GEM. Los archivos se pueden cargar apuntando a los iconos adecuados y pulsando un botón del ratón. Debajo de éste se halla el sistema operativo propio, conocido como AmigaDOS, que hace que la máquina sea auténticamente multitarea. Amiga-





Cuadros de una exposición

La calidad de los gráficos del Commodore Amiga lo colocan muy por encima de otras máquinas de su misma escala de precios. La compleja animación y sus gráficos en alta resolución con sutil coloreado hacen del Amiga una máquina recreativa por excelencia. Es probable que su entorno operativo WIMP estilo Macintosh atraiga a muchos usuarios no especializados, tales como diseñadores gráficos y artistas



El Amiga opera a través de una interface icónica, conocida como Workbench. Están presentes las conocidas ventanas, iconos de disco y de cubo de desperdicios

asignó la tarea de adaptar el Tripos y convertirlo en un sistema operativo funcional para el Amiga en el plazo de un mes. De este modo, Metacomco se comprometió en el proyecto Amiga, produciendo el AmigaBASIC (el intérprete de BASIC empaquetado con la máquina), PASCAL y LISP. Asimismo, ha escrito un macroensamblador y sistemas de desarrollo que se ejecutan bajo Unix y MS-DOS. Si bien el Amiga DOS no soporta conexión en red, el Tripos es fundamentalmente un OS de red; los planes apuntan hacia futuras versiones del AmigaDOS que permitan la conexión en red de hasta 255 Amigas. Para Tim King, este tipo de sistema es una alternativa a las facilidades de la informática de ordenadores de unidad central. La red es suficientemente flexible para aceptar máquinas extras cuando sea necesario, cada una de ellas capaz de utilizar los periféricos de otra estación o de acceder a un "servidor" de archivos centralizado equipado con un disco rígido



Sistema de archivos

El sistema de archivos que utiliza el AmigaDOS es inusual. Se basa en una estructura arborescente con un complejo conjunto de punteros hacia adelante y hacia atrás que unen los bloques del disco. No existe ninguna pista de directorio como tal, sino un bloque "raíz" con punteros a títulos de archivos u otros directorios.

El título de archivo contiene una serie de punteros a cada bloque de datos del archivo y otra información sobre el archivo (como su tamaño expresado en bytes y el momento en que se accedió al mismo por última vez). Los bloques de datos también están encadenados entre sí mediante una serie de punteros hacia adelante y cada bloque de datos también apunta hacia atrás, al título de archivo al cual pertenece. Este complejo sistema de punteros tiene una implicación muy importante. Si un disco se corrompe, a partir de apenas un único bloque "bueno" del disco se puede recuperar la totalidad de la estructura de archivos del disco, siguiendo y rehaciendo los enlaces de punteros entre los bloques. La integridad de los datos es de gran importancia, en particular para los usuarios de gestión, y la capacidad del Amiga para recuperar discos estropeados probablemente contribuirá a aumentar su atractivo

DOS, a su vez, hace llamadas a Intuition, la parte del firmware que se encarga del control de ventanas y ratón.

En la práctica, las capacidades para multitarea del Amiga permiten ejecutar varias aplicaciones de forma simultánea y a la vez independiente. Es probable que éste sea uno de los principales argumentos de venta para el sector de gestión, porque se trata del primer micro de su escala de precio que proporciona tales facilidades. Debido a que las capacidades de proceso de la máquina están compartidas sobre una base de "repartición de tiempo" (siendo objeto cada aplicación de cortos intervalos de atención por turno), se produce una proporcional pérdida de velocidad mientras más aplicaciones se ejecuten juntas en multitarea.

En Estados Unidos, el dominio que mantiene IBM en el mercado de gestión y el rechazo de los usuarios de ordenadores de oficina a la idea de probar máquinas nuevas (a ello se debe la abundancia de clonos IBM existente en el mercado), plantean un grave problema a los fabricantes de ordenadores que desean introducir sus productos en dicho mercado. Aunque el Amiga supera fácilmente al IBM PC por un precio cercano a la mitad, Commodore ha intentado también asegurarse el éxito del Amiga produciendo una unidad de disco de 5½ pulgadas y una opción de software de emulación que hacen que la máquina sea compatible con el IBM. Commodore sostiene que el mismo permite que el Amiga ejecute paquetes tales como el *Lotus 1-2-3*. La técnica de emulación, en esencia, traduce opcodes 8088 a opcodes 68000. Evidentemente, este proceso reduce la velocidad del Amiga, pero existen planes para producir una placa de emulación de hardware que solucionaría este problema.

El Amiga sienta nuevos estándares desde el punto de vista de velocidad, gráficos y sonido, y quizá esté ampliamente justificada la afirmación de su fabricante en el sentido de que su innovador enfoque le hará ganar nuevos mercados.

El Amiga podría catalogarse como una máquina de juegos excelente pero cara, o como una máquina de gestión potente y económica. De modo que, en algún sentido, el ordenador podría experimentar una crisis de identidad, cuya resolución dependería de la respuesta que obtuviera en el mercado.

Commodore Amiga

DIMENSIONES
444 × 300 × 120 mm

CPU
Motorola 68000, operando a 8 MHz

MEMORIA
256 Kbytes, ampliables a 8 Mbytes

PANTALLA
Su resolución máxima para textos es de 80 × 25 caracteres. Las cuatro modalidades para gráficos disponibles van desde una modalidad de 320 × 200 pixels a 32 colores, hasta una modalidad de 640 × 400 pixels a 16 colores

INTERFACES
Dos puertas para ratón, bus de ampliación, interface RS232, interface Centronics, puerta para segunda unidad de disco, puertas para video compuesto y RGB

TECLADO
82 teclas, incluyendo 10 teclas de función y teclado numérico

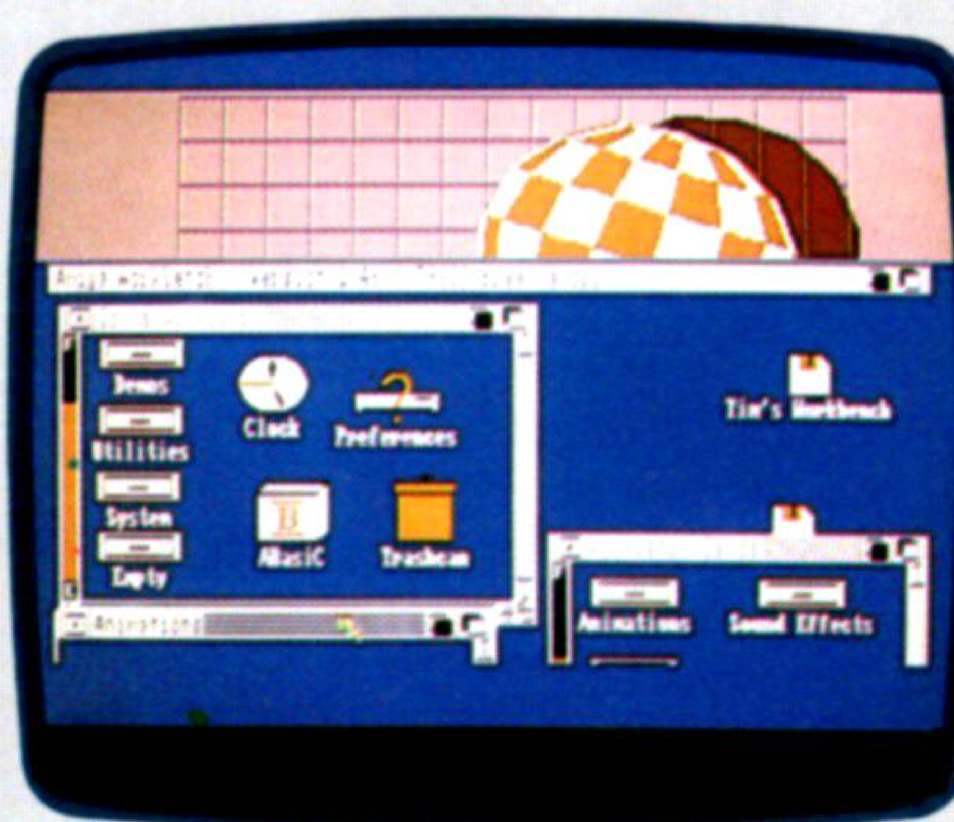
VENTAJAS
Permite la auténtica consecución de multitareas, y las capacidades de sonido y gráficos son sobresalientes

DESVENTAJAS
Gran parte del software prometido para la máquina aún no se ha materializado. Además, Commodore aún no ha decidido a qué mercado dirigirá la máquina. La conquista del público, y, en consecuencia, el éxito del Amiga, dependerán de la política de comercialización

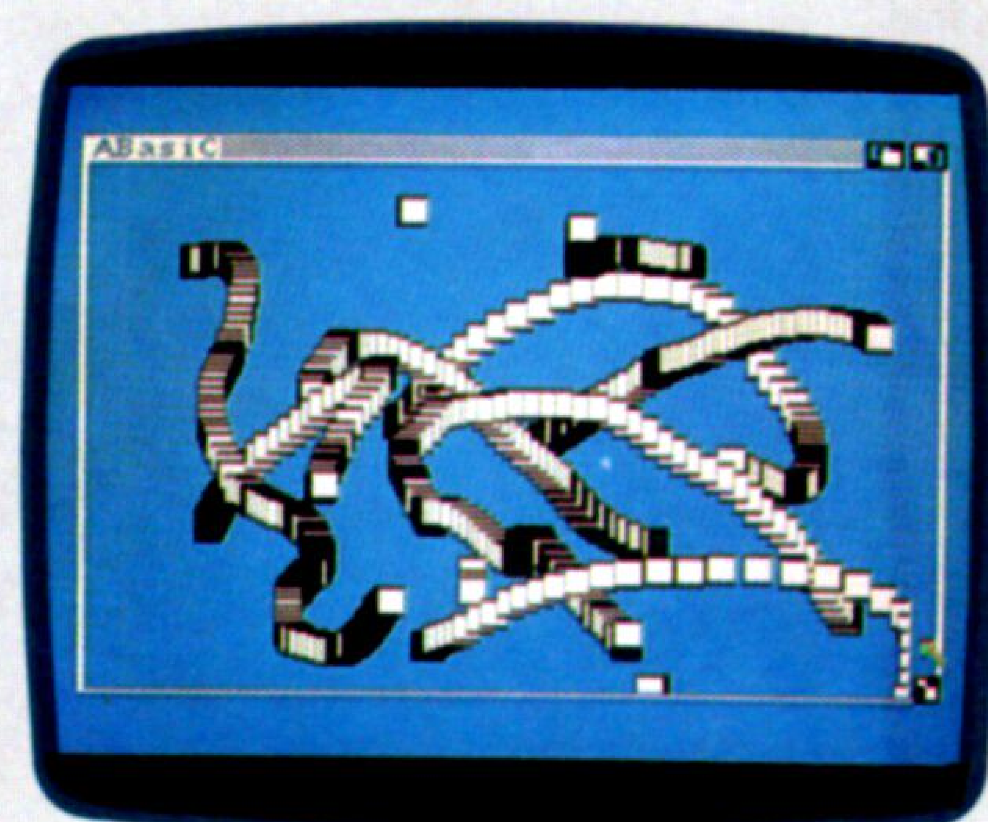
Marcus Wilson-Smith



Los personajes móviles de esta escena animada se crean utilizando objetos *blitter*, "sprites" de software que permiten complejas definiciones de forma y color. Los objetos *blitter* incluyen detección de proximidad. Aquí, la figura de la boca de riego tensa sus músculos para mantener a raya al perro que se acerca



Como se demuestra aquí, el Amiga es un micro multitarea. Podemos ver ejecutándose juntos al Workbench y una demostración de una pelota rebotando



Si bien el AmigaBASIC no es una implementación excelente del lenguaje, sí soporta la mayoría de las facilidades de la máquina. Fue suficiente un programa en BASIC de cinco líneas para producir esta visualización garabateada con el ratón

Hagan sus apuestas

Finalizamos nuestro proyecto añadiéndole al juego las rutinas de apuestas e incluyendo una pantalla de títulos

Apostar al dar vuelta los naipes es una parte integral del juego del veintiuno (o *pontoon*). El programa le concede una cantidad inicial de £10 000. El restante capital del jugador se retiene a lo largo del juego en la variable SK. Existen varios métodos de apuesta. Adoptaremos el siguiente sistema:

- El jugador debe apostar £50 al comienzo de cada ronda para entrar en el juego.
- Esta apuesta inicial la realiza automáticamente el programa.

Habiéndosele repartido un naipe, el jugador puede "comprar" un segundo naipe haciendo una apuesta adicional de hasta £1 000. Un jugador perspicaz reconocerá que algunos naipes (un as, p. ej.) son más prometedores que otros y apostará en consecuencia.

Al jugador se le pueden repartir naipes adicionales hasta que se plante o se pase. Si el jugador cree que posee una mano sumamente favorable, o si apuesta demasiado bajo en su primer naipe, entonces puede optar por doblar su apuesta y se le repartirá otro naipe más.

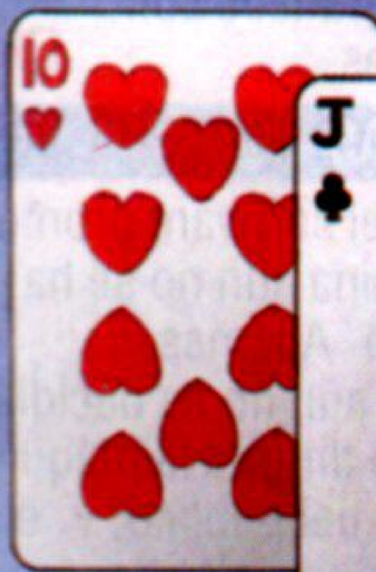
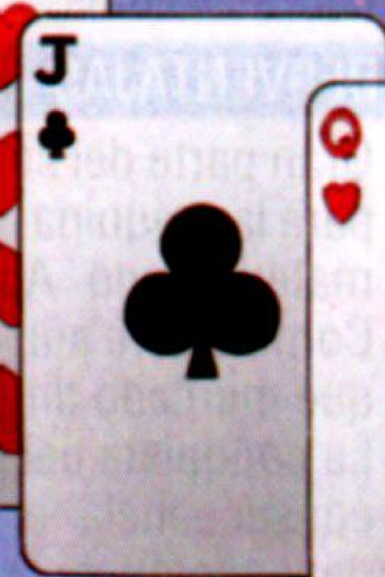
El jugador gana la ronda si, después de que el ordenador (la banca) juega su mano, su marcador resulta superior al de la banca. En este caso su apuesta se le devuelve y se le suma a su capital una cantidad adicional igual a su apuesta. De lo contrario, pierde el dinero apostado.

Estrategias de apuesta

Tal como sucede con todas las apuestas, las estrategias de éxito se basan en el principio de "ganar mucho y perder poco". La fase crucial de la apuesta en nuestra versión del veintiuno se identifica con el momento en que el jugador adquiere un segundo naipe. Abajo esbozamos algunas sugerencias sencillas para mejorar su rendimiento

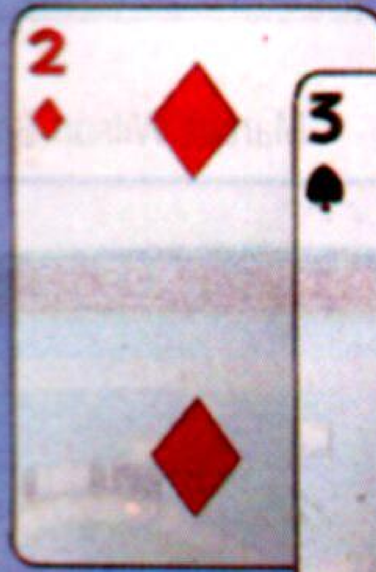
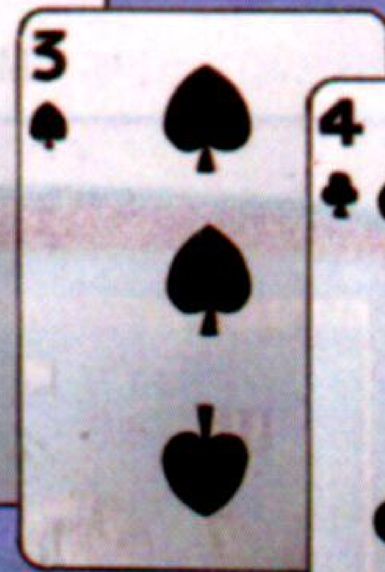
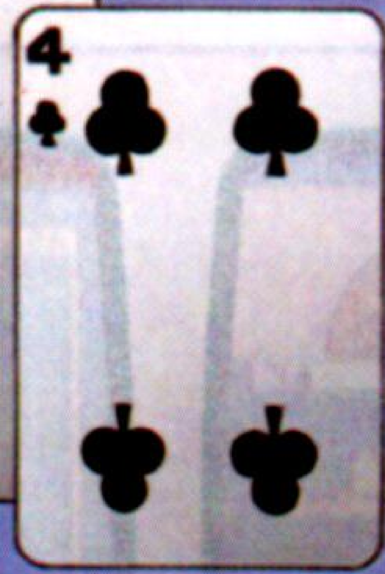


Vale un premio muy bueno
- Hasta 16 naipes en el mazo que den *royal pontoon/ pontoon*
- El as se cuenta como 1 u 11, valor flexible a medida que se desarrolla la mano
- Posible juego de cinco naipes

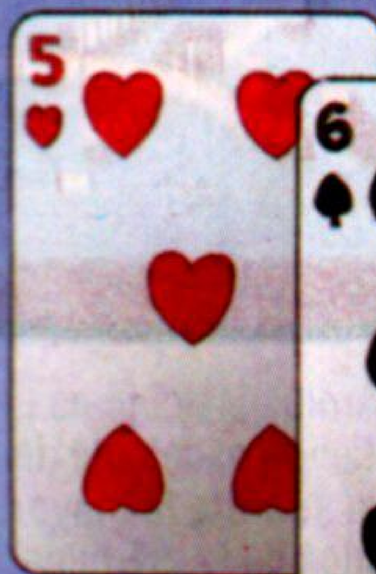





Vale un buen premio
- Subsiguientes 2, 3 o 4 permiten quemar los naipes
- Un segundo naipe alto (9, 10, J, Q, K) permite plantarse
- Un ulterior as da *pontoon/royal pontoon*
- El inconveniente está en que una ulterior carta de "orden mediano" (5 o 6) puede hacer que el tercer naipe haga pasarse la mano

Vale un premio mediano
- Los naipes bajos dan posibilidades de juego de cinco naipes
- Posibilidad de pasarse si el segundo naipe es alto





Sólo un premio bajo
- Los naipes de orden mediano pueden plantear problemas si se reparte un 9, 10, J, Q o K como segundo naipe

Kevin Jones

El pote de las apuestas

A lo largo del juego se visualizan en la pantalla la apuesta actual y los niveles de capital inicial restantes. Los títulos para este "pote" los imprime la rutina de la línea 4200, que es llamada por la rutina "inicializar el juego" de la línea 625. Esta línea también restablece una variable, BT, que se utiliza para llevar el registro de la apuesta efectuada durante la ronda actual.

La rutina "imprimir premio", que empieza en la línea 4300, es una rutina de propósito general a la que se puede llamar desde varios puntos del programa principal. Se realizan apuestas adicionales estableciendo la variable SB antes de llamar a la rutina. En circunstancias normales, SB se suma a la apuesta acumulativa, BT, y se resta del paquete inicial restante, SK. No obstante, como esta rutina es general, necesita realizar varias comprobaciones sobre la validez de la apuesta a efectuar. Si al jugador le quedan menos de £50, y se efectúa la apuesta automática inicial (indicada porque la bandera BG=1), la rutina imprimirá un mensaje anunciando que los fondos que quedan son insuficientes y da al jugador la opción de recomenzar el juego desde el principio.

Si el jugador desea doblar su apuesta, pero no dispone de capital suficiente, entonces se debe imprimir un mensaje y desautorizar la apuesta. Esta comprobación se efectúa restando la apuesta proyectada y viendo si queda un paquete inicial restante negativo, SK. De ser así, entonces se vuelve a sumar la apuesta y se sale de la rutina.

La tercera circunstancia inusual se produce si el jugador intenta comprar un segundo naipe apostando más dinero del que posee. En este caso, la rutina imprimirá un mensaje apropiado y reducirá la apuesta adicional a la cantidad de capital que le quede disponible al jugador.

La rutina pasa luego a borrar la apuesta impresa previamente y los valores de paquete inicial restante sobreimprimiendo algunos espacios antes de imprimir los nuevos valores.



La apuesta inicial de £50 se realiza automáticamente al comienzo de cada ronda mediante la línea 72. Las líneas 73-80 permiten que el jugador compre un segundo naipe, comprobando la cantidad entrada para asegurar que no supere el límite de £1 000.

Cuando el jugador pide para recibir más naipes en la rutina que comienza en la línea 2700, puede optar por doblar en su último naipe. La rutina para doblar propiamente dicha se halla en la línea 2900 y llama a la rutina "imprimir apuesta". Si el jugador no puede permitirse doblar (lo que se indicaría si CA=1), entonces se sale de la rutina y el programa revierte a la rutina pedir/plantarse normal. De lo contrario, se reparte al jugador un último naipe, se evalúa la mano y termina la rutina.

En el capítulo anterior de este proyecto desarrollamos las rutinas que deciden qué mano ha ganado y luego imprimen un mensaje "ganar o perder", según convenga. Sólo resta añadir una línea que sume la cantidad apostada al total de paquete inicial restante en el caso de que el jugador gane la

Rutinas de apuestas

BBC Micro

```
  30 GOSUB 4000
  72 BG=1:SB=IS:GOSUB 4300:BG=0
  73 GOSUB 700:PRINT"COMPRAS UN NAIPE (S/N)";
  74 RESP$=GET$
  75 IF RESP$<> CHR$(13) THEN PRINT RESP$
  77 IF RESP$<> "S" THEN 85
  78 RESP$="":GOSUB 700:INPUT "TU APUESTA (MAX
     £1000)"RESP$
  79 IF VAL(RESP$)>1000 THEN 78
  80 SB=VAL(RESP$):GOSUB 4300
 210 GOSUB 700:PRINT"GANAS £";BT
 220 SB=-2* BT:BT=-SB:GOSUB 4300
 565 SK=10000:IS=50
 610 HP(1)=1:HP(2)=1
 625 BT=0:GOSUB 4200
2740 IF RESP$="D" THEN GOSUB 2900:IF CA=0 THEN
     RETURN
2900 REM **** DOBLAR ****
2910 DB=1:SB=BT:GOSUB 4300
2915 IF CA=1 THEN DB=0:RETURN
2920 FL=0:PL=1:GOSUB 1300
2930 GOSUB 800
2940 DB=0:RETURN
4000 REM
4010 CLS
4020 TX=13:TY=3:GOSUB 900:PRINT"BBC/ELECTRON"
4030 PRINT TAB(TX+3);"PONTOON"
4040 PRINT TAB(TX+5);"POR"
4050 PRINT TAB(TX-7);"PETE SHAW & STEVE COLWILL"
4060 TX=9:TY=8:GOSUB 900:PRINT"TE QUEDAN £";SK
4070 TX=10:TY=12:GOSUB 900:PRINT"PULSA CUALQUIER
     TECLA PARA JUGAR"
4080 A$=GET$
4090 RETURN
4200 REM
4210 COLOUR 4:TX=24:TY=18:GOSUB 900:PRINT"TU
     APUESTA"
4220 TX=24:TY=20:GOSUB 900:PRINT"INICIAL RESTANTE"
4230 RETURN
4300 REM
4302 CA=0:REM NO SE PUEDE PERMITIR DOBLAR
4305 IF SK>=50 OR BG=0 THEN 4310
4306 RESP$="":GOSUB 700:INPUT"TE HAS QUEDADO SIN
     DINERO!" VUELVES A JUGAR (S/N)";RESP$
4307 IF RESP$="S" THEN RUN
4308 END
4310 SK=SK-SB:BT=BT+SB
4315 IF DB=1 AND SK< 0 THEN GOSUB 700:PRINT"NO TE LO
     PUEDES PERMITIR!"
4317 IF DB=1 AND SK< 0 THEN BT=BT-SB:CA=1:RETURN
4320 IF SK< 0 THEN BT=SK+SB:SK=0:GOSUB
     700:PRINT"SOLO PUEDES PERMITIRTE £";BT
4340 COLOUR 1:TX=24:TY=19:GOSUB 900:PRINT
     LEFT$(SP$,15)
4345 TX=24:TY=19:GOSUB 900:PRINT "£";BT
4350 TX=24:TY=21:GOSUB 900:PRINT LEFT$(SP$,15)
4355 TX=24:TY=21:GOSUB 900:PRINT "£";SK
4360 RETURN
```

Gama Amstrad CPC

```
  30 GOSUB 400:REM pantalla de titulos
  72 bg=1:sb=is:GOSUB 4300:bg=0:REM imprimir apuesta
  73 GOSUB 700:PRINT "compras un naipe (s/n)";
  74 resp$="":While resp$="":resp$=INKEY$:WEND
  75 IF resp$<>CHR$(13) THEN PRINT resp$
  77 IF resp$<>"s" THEN 85
  78 resp$="":GOSUB 700:INPUT"tu apuesta (max
     £1000)";resp$
  79 IF VAL(resp$)>1000 THEN 78
  80 sb=VAL(resp$):GOSUB 4300:REM imprimir apuesta
 210 GOSUB 700:PEN negro:PRINT"ganas £";:PEN
     blanco:PRINT bt
 220 sb=-2* bt:bt=-sb:GOSUB 4300:REM imprimir apuesta
 565 sk=10000:is=50:REM paquetes
 610 hp(1)=1:hp(2)=1
 625 bt=0:sb=0:GOSUB 4200:REM imprimir pote apuestas
2740 IF resp$="d" THEN GOSUB 2900:IF ca=0 THEN RETURN
2900 REM **** doblar ****
2910 db=1:sb=bt:GOSUB 4300:REM imprimir apuesta
2915 IF ca=1 THEN db=0:RETURN
2920 fl=0:pl=0:GOSUB 1300:REM repartir
2930 GOSUB 800:REM evaluar
2940 db=0:RETURN
4000 REM **** titulo etc ****
4010 CLS
4020 tx=11:ty=3:GOSUB 900:PEN rojo:PRINT"Gama Amstrad
     CPC"
4030 PRINT TAB(tx+6)"Pontoon"
4040 PRINT TAB(tx+8)"Por"
4050 PRINT TAB(tx+3)"Steve Colwill"
4060 tx=9:ty=8:GOSUB 900:PEN negro:PRINT"Tu paquete
     inicial es de £";sk
4070 tx=10:ty=12:GOSUB 900:PEN blanco:PRINT"Pulsa una
     tecla para jugar"
4080 WHILE INKEY$="":WEND
4090 RETURN
4200 REM **** pote de apuestas ****
4210 tx=24:ty=18:GOSUB 900:PEN rojo:PRINT"Tu apuesta"
4220 tx=24:ty=20:GOSUB 900:PRINT"Paquete restante"
4230 RETURN
4300 REM **** imprimir paquete ****
4302 ca=0:REM no se puede permitir doblar bandera
4305 IF sk> =50 OR bg=0 THEN 4310
4306 resp$="":GOSUB 700:PEN rojo:INPUT"Te has quedadc
     sin dinero! Vuelves a jugar (s/n)";resp$
4307 IF resp$="s" THEN RUN
4308 END
4310 sk=sk-sb:bt=bt+sb
4315 IF db=1 AND sk<0 THEN GOSUB 700:PEN
     rojo:PRINT"No puedes permitirtelo!":bt=bt-sb:sk=
     sk+sb:ca=1:RETURN
4320 IF sk<0 THEN bt=sk+sb:sk=0:GOSUB 700:PEN
     rojo:PRINT"Solo puedes permitirte £";bt
4340 tx=24:ty=19:GOSUB 900:PRINT SPACE$(15)
4345 tx=24:ty=19:GOSUB 900:PEN blanco:PRINT"£";bt
4350 tx=24:ty=21:GOSUB 900:PRINT SPACE$(15)
4355 tx=24:ty=21:GOSUB 900:PRINT"£";sk
4360 RETURN
```

mano. Esto se realiza insertando la línea 220 en el bucle principal.

Ahora hemos completado los listados para cada una de las cuatro máquinas y el jugador podrá jugar al juego completo contra el ordenador. El programa barajará el mazo automáticamente al comienzo del juego y pulsando SHIFT/S (SYMBOL SHIFT/S en el Spectrum) se puede conseguir que lo vuelva a barajar automáticamente después de cada ronda.

En un proyecto futuro usted podría mejorar la visualización de naipes y las rutinas de reparto dadas y crear un juego del veintiuno para más de un jugador.

Sinclair Spectrum

```
30 GO SUB 4000: REM PANTALLA DE TITULOS
72 LET BG=1: LET SB=IS:GO SUB 4300: LET BG=0: REM
   IMPRIMIR APUESTA
73 LET A$="": GO SUB 700: PRINT INK 2;"COMPRAS UN
   NAIPE (S/N) ";
74 LET A$=INKEY$: IF $="" THEN GO TO 74
75 IF A$<> CHR$ 13 THEN PRINT A$
77 IF A$<> "S" THEN GO TO 85
78 LET A$="": GO SUB 700: INPUT "TU APUESTA (MAX
   £1000)";LINE A$
79 IF VAL A$> 1000 THEN GO TO 78
80 LET SB=VAL A$: GO SUB 4300: REM IMPRIMIR
   APUESTA
210 GO SUB 700: PRINT "GANAS £";BT
220 LET SB=-2*BT: LET BT=-SB: GO SUB 4300: REM
   IMPRIMIR APUESTA
565 LET SK=10000: LET IS=50: REM PAQUETES
610 LET P(1)=1:LET P(2)=1
625 LET BT=0:GO SUB 4200:REM IMPRIMIR POTE
   APUESTAS
2740 IF A$="D" THEN GO SUB 2900: IF CA=0 THEN RETURN:
   REM DOBLAR
2900 REM **** DOBLAR ****
2910 LET DB=1: LET SB=BT: GO SUB 4300: REM IMPRIMIR
   APUESTA
2915 IF CA=1 THEN LET DB=0: RETURN
2920 LET FL=0: LET PL=1: GO SUB 1300:REM REPARTIR
2930 GO SUB 800:REM EVALUAR
2940 LET DB=0: RETURN
4000 REM **** TITULO ETC ****
4010 CLS
4020 LET TX=10: LET TY=3: GO SUB 900: PRINT " ZX
   SPECTRUM"
4030 PRINT TAB(TX+3);"PONTOON"
4040 PRINT TAB (TX+5);"POR"
4050 PRINT TAB(TX-7);"PETE SHAW & STEVE COLWILL"
4060 LET TX=5:LET TY=8: GO SUB 900: PRINT "TU PAQUETE
   ES DE £";SK
4070 LET TX=5: LET TY=12:GO SUB 900: PRINT FLASH
   1;"PULSA CUALQUIER TECLA PARA JUGAR"
4080 LET A$=INKEY$: IF A$="" THEN GO TO 4080
4090 RETURN
4200 REM **** IMPRIMIR PAQUETE ****
4210 PRINT AT 18,15;"TU APUESTA"
4220 PRINT AT 20,15;"PAQUETE RESTANTE"
4230 RETURN
4300 REM **** IMPRIMIR PAQUETE ****
4302 LET CA=0: REM NO SE PUEDE PERMITIR DOBLAR
   BANDERA
4305 IF SK> =50 OR BG=0 THEN GO TO 4310
4306 LET A$="": GO SUB 700: INPUT "TE HAS QUEDADO SIN
   DINERO! VUELVES A JUGAR (S/N)";A$
4307 IF A$="S" THEN RUN
4308 STOP
4310 LET SK=SK-SB: LET BT=BT+SB
4315 IF DB=1 AND SK<0 THEN GO SUB 700: PRINT FLASH
   1;"NO PUEDES PERMITIRTELO!"
4317 IF DB=1 AND SK<0 THEN LET BT=BT-SB: LET
   SK=SK+SB: LET CA=1: RETURN
4320 IF SK<0 THEN LET BT=SK+SB: LET SK=0: GOSUB 700:
   PRINT FLASH 1; "SOLO PUEDES PERMITIRTE £";BT
4340 LET TX=24: LET TY=18: GO SUB 900: PRINT S$(TO 15)
4345 LET TX=24: LET TY=18: GO SUB 900: PRINT "£";BT
4350 LET TX=24: LET TY=20: GO SUB 900:PRINT S$(TO 15)
4355 LET TX=24: LET TY=20: GO SUB 900: PRINT "£";SK
4360 RETURN
```

Commodore 64

```
72 BG=1:SB=IS:GOSUB 4300:BG=0:REM IMPRIMIR
   APUESTA
73 RESP$="":GOSUB 700:PRINT"COMPRAS UN NAIPE
   (S/N)";
74 GET RESP$:IF RESP$="" THEN 74
75 IF RESP$<>CHR$(13) THEN PRINT RESP$
77 IF RESP$<>"S" THEN 85
78 RESP$="":GOSUB 700:INPUT"TU APUESTA (MAX
   £1000)";RESP$
79 IF VAL(RESP$)>1000 THEN 78
80 SB=VAL(RESP$):GOSUB 4300:REM IMPRIMIR APUESTA
210 GOSUB 700:PRINT CHR$(156);"GANAS £";CHR$(5);BT
220 SB=-2* BT:BT=-SB:GOSUB 4300:REM IMPRIMIR
   APUESTA
565 SK=10000 : IS=50:REM PAQUETES
610 HP(1)=1:HP(2)=1
625 BT=0:GOSUB 4200:REM IMPRIMIR POTE DE APUESTAS
2740 IF RESP$="D" THEN GOSUB 2900: IF CA=0 THEN
   RETURN
2900 REM **** DOBLAR ****
2910 DB=1:SB=BT:GOSUB 4300:REM IMPRIMIR APUESTA
2915 IF CA=1 THEN DB=0:RETURN
2920 FL=0:PL=1:GOSUB 1300:REM REPARTIR
2930 GOSUB 800:REM EVALUAR
2940 DB=0:RETURN
4000 REM **** TITULO ETC ****
4010 PRINT CHR$(147):REM LIMPIAR PANTALLA
4020 TX=13:TY=3:GOSUB 900:PRINT
   CHR$(156);"COMMODORE 64"
4030 PRINTTAB(TX+3);"PONTOON"
4040 PRINTTAB(TX+5);"POR"
4050 PRINTTAB(TX);"STEVE COLWILL"
4060 TX=9:TY=8:GOSUB 900:PRINT CHR$(28);"TU PAQUETE
   INICIAL ES DE £";SK
4070 TX=10:TY=12:GOSUB 900:PRINT CHR$(5);"PULSA UNA
   TECLA PARA JUGAR"
4080 GET A$:IF A$="" THEN 4080
4090 RETURN
4200 REM **** POTE DE APUESTAS ****
4210 TX=24:TY=18:GOSUB 900:PRINT CHR$(156);"TU
   APUESTA"
4220 TX=24:TY=20:GOSUB 900:PRINT"PAQUETE
   RESTANTE"
4230 RETURN
4300 REM **** IMPRIMIR PAQUETE ****
4302 CA=0:REM NO SE PUEDE PERMITIR DOBLAR BANDERA
4305 IF SK> =50 OR BG=0 THEN 4310
4306 RESP$="":GOSUB 700:PRINT CHR$(28);:INPUT"TE HAS
   QUEDADO SIN DINERO! VUELVES A JUGAR (S/N)";RESP$
4307 IF RESP$="S" THEN RUN
4308 END
4310 SK=SK-SB:BT=BT+SB
4315 IF DB=1 AND SK< 0 THEN GOSUB 700:PRINT
   CHR$(28);"NO PUEDES PERMITIRTELO!"
4317 IF DB=1 AND SK< 0 THEN
   BT=BT-SB:SK=SK+SB:CA=1:RETURN
4320 IF SK< 0 THEN BT=SK+SB:SK=0:GOSUB 700:PRINT
   CHR$(28);"SOLO PUEDES PERMITIRTE £";BT
4340 TX=24:TY=19:GOSUB 900:PRINT LEFT$(SP$,15)
4345 TX=24:TY=19:GOSUB 900:PRINT CHR$(5);"£";BT
4350 TX=24:TY=21:GOSUB 900:PRINT LEFT$(SP$,15)
4355 TX=24:TY=21:GOSUB 900:PRINT CHR$(5);"£";SK
4360 RETURN
```



Programa pionero

"Shadow of the unicorn" (La sombra del unicornio) es el primer programa de juegos que incorpora un dispositivo que impide la acción de los piratas

Combatir la piratería de las cintas se ha convertido en una obsesión para las casas de software. Las empresas reconocen estar perdiendo millones de pesetas al año a causa de los piratas. El obstáculo que impide que la cruzada en contra de los piratas llegue a buen puerto reside en que el soporte más popular para los juegos por ordenador sigue siendo la cinta de cassette, que, lamentablemente para las casas de software, es la más fácil de copiar.

La idea que subyace en el *dongle*, un dispositivo de software que se enchufa en el ordenador, ha estado rondando en la mente de los fabricantes de software durante un tiempo. El *dongle* contiene parte de un programa, y sin el mismo el resto de éste (retenido en cassette o disco) no se ejecutará. Pero se presenta un grave inconveniente: los costos de fabricación de los *dongles* son más altos que los de las cassettes estándares. Esto significa que las casas de software que adopten el sistema deben convencer a los usuarios de que los *dongles* son para proteger mejor los intereses de éstos y no sólo para beneficiar exclusivamente al fabricante.

Mikro-Gen es la primera casa de juegos que adopta el *dongle* como método para impedir la piratería de software. El primer juego que lanza bajo el nuevo formato es *Shadow of the unicorn* (La sombra del unicornio), que está dividido entre un programa retenido en una cinta de cassette normal y un dispositivo *dongle* que se enchufa en la puerta para ampliación del Spectrum.

El *dongle* propiamente dicho consta de una EPROM de 16 Kbytes, una puerta para palanca de mando y un chip decodificador de palanca de mando. La EPROM contiene las facilidades de carga para la cassette y varias de las rutinas gráficas utilizadas en el juego. Tras el encendido, la EPROM se asocia con las direcciones de las zonas de memoria normalmente ocupadas por la ROM de BASIC, permitiendo espacio adicional para el resto del programa.

Aventura de estilo recreativo (con reminiscencias de *Lords of midnight*), el juego se desarrolla en los reinos de Oronfal y Falforn, y la trama consiste en derrotar a las fuerzas del mal que ahora habitan en el territorio. Inicialmente, el jugador controla a tres personajes: Mithulin (el rey de Oronfal), Ulin-Gail (un sátiro) y Avarath (un hechicero). Sin embargo, a lo largo del juego hay otros personajes que el usuario puede controlar una vez que los encuentra. El primero de éstos es Holdin, el capitán de Falforn, que comienza el juego en la misma posición que el hechicero y, por tanto, queda inmediatamente bajo el control del jugador.

Como en otras aventuras recreativas, no sólo hay varios escenarios a explorar, sino también enemigos a vencer y objetos a recoger que pueden ayudar al jugador a medida que el juego avance. En las primeras etapas, los enemigos más comunes son amenazadoras y agresivas criaturas con aspecto de duendes, quienes pueden ser destruidas fácilmente por algunos de los personajes, como el hechicero. Otros personajes sólo pueden eliminarlos con gran dificultad, mientras que el resto, que no dispone de armas, sólo puede huir ante el ataque de los duendes.

Mientras se controla a un personaje determinado, los factores de "energía" y "daño del jugador" se visualizan en forma de barras en la parte superior de la pantalla. Evidentemente, el ataque de un duende reducirá la energía del personaje y, si se permite que el duende alcance al personaje, aumentará el factor de daño. La energía se puede reponer alimentándose con uno de los arbustos distribuidos por el reino. Los daños, sin embargo, por lo general sólo se repararán con el tiempo.

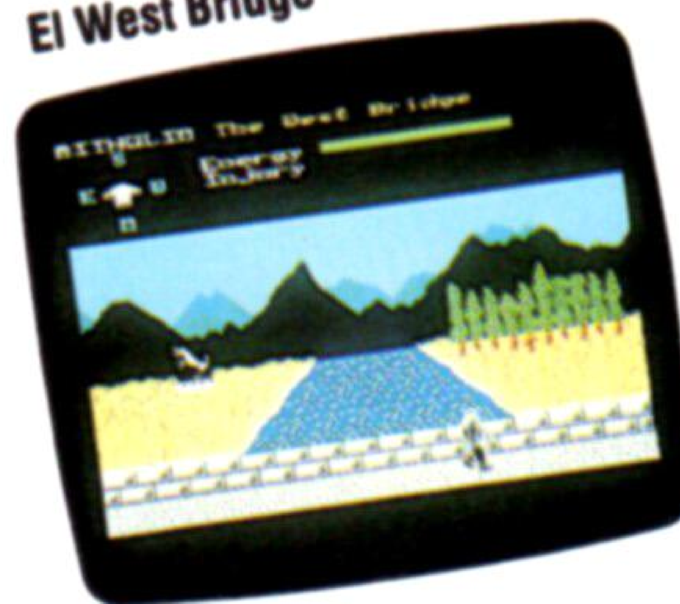
A medida que los personajes se desplazan por el reino, se encuentran con numerosos edificios en los que experimentarán dificultades para introducirse. Intentar moverse en la dirección de la puerta sólo hará que la escena cambie, dejando al personaje al otro lado del edificio. Como cabe esperar, existen formas y medios de entrar en los edificios, pero sólo si el acercamiento se realiza del modo correcto.

Al igual que otros juegos de aventuras, *Shadow of the unicorn* viene acompañado de un libro que ofrece detalles sobre el ambiente del juego y proporciona al jugador variadas pistas acerca de cómo completar la aventura.

Shadow of the unicorn es, sin duda alguna, una jugada audaz por parte de Mikro-Gen. Pero es evidente que la empresa ha hecho un esfuerzo por presentar el juego al usuario de la manera más atractiva posible. Queda por ver si conseguirá redefinir el mercado y asestar el golpe de gracia a los piratas.

Shadow of the unicorn (La sombra del unicornio):
Para el Sinclair Spectrum
Editado por: Mikro-Gen, Unit 15, Western Centre, Bracknell, Berks, Gran Bretaña
Autor: Dale McLoughlin
Palanca de mando: Opcional
Formato: Cassette y EPROM

El West Bridge



Cerca de Olindel



Siga por aquí
Si bien *Shadow of the unicorn* tiene incluidos en su diseño numerosos elementos recreativos, la base del juego es la de una aventura tradicional. El jugador controla a algunos de los personajes, quienes buscan en los diversos escenarios las pistas y ayudas que les permitirán vencer a las fuerzas del mal



Paquetes con valor añadido
El *dongle* de Mikro-Gen, que se suministra con la aventura *Shadow of the unicorn*, le proporciona al jugador 16 Kbytes adicionales de memoria utilizable, mejorando de este modo la cantidad y calidad del juego. Con el Mikro-Plus también se suministra una interface para palanca de mando, puesto que el dispositivo no se puede instalar con la Interface 2

Influencia paterna

Examinaremos la estructura arborescente en la cual se basa el directorio Unix así como algunas de las instrucciones más importantes

En el capítulo anterior vimos cómo cada usuario de un sistema operativo multiusuario como el Unix posee su propia área especial para almacenamiento de disco denominada *directorio*, en donde pueden conservar sus propios programas y datos protegidos de la interferencia de otros usuarios mediante un sistema de contraseñas. Además, también es necesario tener: "directorios públicos" que contengan los programas del sistema disponibles para todos; "directorios del sistema", que conservan, entre otras cosas, información sobre los usuarios y sus directorios; y la capacidad de presentar más de un directorio de modo que los usuarios puedan mantener separados diferentes aspectos de su trabajo.

Esto significa que pueden coexistir una gran cantidad de directorios, teniendo los usuarios diversos niveles de acceso a los mismos. Por ejemplo, todos los usuarios querrán moverse libremente entre sus propios directorios pudiendo, al mismo tiempo, ejecutar programas de los directorios públicos, pero sólo algunos de ellos gozarán de libertad para introducir cambios en estos programas.

Es bueno saber qué otros directorios hay, e incluso qué archivos hay en ellos. Asimismo, debe ser posible trasladar o copiar archivos entre directorios, aunque no necesariamente poder introducir ningún cambio en ellos. El Unix posee una estructura de directorios que permite que los usuarios creen directorios nuevos y supriman aquellos que se han creado. Además, incluye instrucciones para desplazarse a distintos directorios y transferir archivos.

La estructura de directorios que emplea el Unix es la estructura *arborescente*, utilizada en informática para organizar conjuntos de datos con complejas interrelaciones. Cada directorio puede contener varios archivos y subdirectorios, a todos los cuales se les confieren nombres de modo que puedan ser referenciados. Aunque en muchas situaciones el Unix tratará los subdirectorios y los archivos indistintamente, sin embargo lleva el registro de cuál es cuál. Tal subdirectorio es un directorio *descendiente*, o *hijo*, del directorio *padre* que retiene su nombre. Cada directorio, por consiguiente, tendrá un padre y una cantidad no especificada de descendientes.

Si ascendemos en la cadena consultando cada vez al directorio padre, hemos de llegar finalmente a una interrupción en un directorio sin padre. Se dice que éste es el directorio *raíz* y se alude a él mediante el nombre /. A cualquier otro directorio se puede aludir por su *nombre de camino* (*pathname*), que es una lista de directorios comenzando desde la raíz, separando al nombre de cada directorio con /.

El diagrama ilustra un sistema de directorios Unix simplificado. El nombre completo para el directorio del usuario Juan es /usr/dept1/Juan. Un archivo también tiene un nombre completo, que se compone del nombre de camino hacia el directorio que lo contiene, seguido de /nombrearchivo. Normalmente no es necesario utilizar un nombre de camino completo para especificar un archivo dado; con frecuencia no sólo se lo puede abreviar, sino también omitir por completo cuando se está trabajando exclusivamente con el propio directorio del usuario.

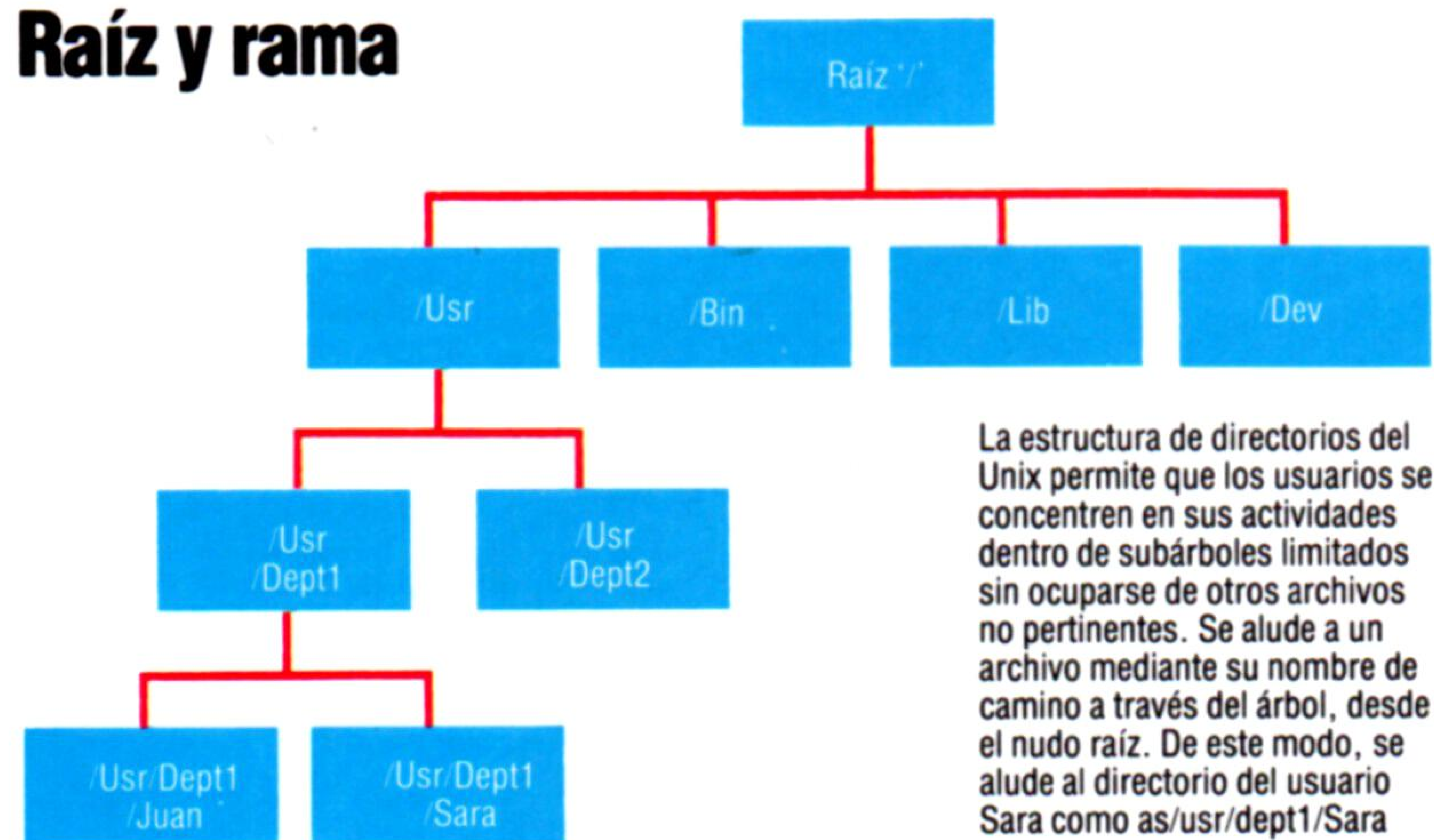
Si el usuario necesita averiguar el nombre de camino completo del directorio en el que se halla, la instrucción pwd (*print working directory*: imprimir el directorio con el cual se está trabajando) se lo proporcionará.

El Unix posee más libertad que casi todos los otros sistemas operativos en cuanto a los nombres que se pueden utilizar para archivos y directorios. Se pueden emplear hasta 14 caracteres, cualesquiera sean, incluyendo espacios; pero es aconsejable evitar ciertos caracteres especiales que el Unix reserva para fines particulares. Éstos son:

\/ " ' *; – ? [] () ¯ ! ${}<> :

Cuando se alude a nombres de archivos y directorios, el Unix tiene un sistema de caracteres de más-

Raíz y rama



La estructura de directorios del Unix permite que los usuarios se concentren en sus actividades dentro de subárboles limitados sin ocuparse de otros archivos no pertinentes. Se alude a un archivo mediante su nombre de camino a través del árbol, desde el nudo raíz. De este modo, se alude al directorio del usuario Sara como as/usr/dept1/Sara

Caroline Clayton

Opciones de lista

Junto con el comando ls se puede utilizar lo siguiente:

–a	Lista todas las entradas, incl. arch. del sist.
–c	Lista por tiempo de creación de archivo
–l	Listado completo
–m	Salida en flujo separada por comas
–r	Orden inverso
–s	Dando el tamaño en bloques
–F	Marca los directorios con un / y los programas ejecutables con *
–R	Lista el cont. de todos los subdirectorios

Las opciones se pueden combinar (–a y –r, p. ej., se pueden unir para obtener –ar)



Diálogo con el Unix

```
Berkeley 4.2 Vox/Unix (infsc3)
Type <Ctrl-D> to disconnect

login:com—mcc
Password:
{observe que la contraseña no se reproduce en la pantalla}

You are a Normal user (Class 3)
Jobs: 19 Superiors: 2 Maximum: 21
Last login: Fri Oct 18 11:45:37 on ttyn05

    Welcome to the Information Sciences VAX/UNIX System.

%pwd
/mnt/com/com-mcc                {éste es mi directorio base}
%cd fred                        {pasando a un subdirectorio}
%ls                             {listar todos los archivos de este directorio}
rec.c   receive   rx.p   transmit   tx.p
%ls —a
        rec.c   receive   rx.p   transmit   tx.p

{observe la presencia de dos nombres extras '.', que es este directorio, y '..', que es el
padre inmediato del mismo}

%ls –l
total 42
-rw-rw-r--    1 com-mcc      502 Sep 17 12:07 rec.c
-rwxr-xr-x    1 com-mcc    18432 Oct 21 11:02 receive
-rw-r--r--    1 com-mcc     1068 Oct 18 14:44 rx.p
-rwxr-xr-x    1 com-mcc    19456 Oct 21 11:03 transmit
-rw-r--r--    1 com-mcc     1244 Oct 21 11:01 tx.p

{los tres grupos de 'rwx' indican derechos de acceso para 1. el propietario de este
directorio, 2. otros del grupo del propietario, y 3. todas las otras personas. r significa
lectura permitida, w significa escritura permitida y x significa ejecución permitida. Un
guión significa que no está permitido el acceso de este tipo}

%ls ?x.p                        {utilizando caracteres de máscara}
rx.p   tx.p
%ls r*
rec.c   receive   rx.p
%ls   *.[pc]
rec.c   rx.p   tx.p

%mkdir mike          {haciendo un directorio nuevo}
%ls –l
total 43
drwxr-xr-x    2 com-mcc       24 Oct 21 11:10 mike
-rw-rw-r--    1 com-mcc      502 Sep 17 12:07 rec.c
-rwxr-xr-x    1 com-mcc    19456 Oct 21 11:03 transmit
-rw-r--r--    1 com-mcc     1244 Oct 21 11:01 tx.p

{observe que un subdirectorio nuevo se indica mediante una 'd'.}

%cd mike                  {cambiar directorio de trabajo a uno nuevo}
%pwd
/mnt/com/com-mcc/fred/mike
%cd ..                    {de vuelta al directorio padre}
-rwxr-xr-x    1 com-mcc    18432 Oct 21 11:02 receive
-rw-r--r--    1 com-mcc     1068 Oct 18 14:44 rx.p
%pwd
/mnt/com/com-mcc/fred
%cp rx.p mike             {copiar archivo 'rx.p' en el nuevo directorio}
%mv tx.p mike             {trasladar archivo 'tx.p' al nuevo directorio}
%ls                       {ahora 'tx.p' se ha ido de este directorio}
mike   rec.c   receive   rx.p   transmit
%cd   mike
%ls
rx.p   tx.p
%who                      {averiguar quién más está utilizando el sistema}
root                      console Oct 21 08:07
com-rgd                   ttyn00  Oct 21 09:45
ccs-klf                   ttyn01  Oct 21 09:45
cs4bc                     ttyn04  Oct 21 10:57
com-mcc                   ttyn05  Oct 21 10:58
ccs-mdb                   ttyn06  Oct 21 10:06
cs4cy                     ttyn07  Oct 21 11:06
com-ah                    ttyn08  Oct 21 11:00
com-jh1                   ttyn10  Oct 21 11:05
cs4bg                     ttyn11  Oct 21 11:05
%finger com-mcc           {detalles sobre otro usuario}
Login name: com-mcc       Real name: curtis
Department: Not known
Directory: /mt/com/com-mcc
Shell: /bin/csh
Logged in at 10:58 on Mon Oct 21 on ttyn05 36 secs idle No Plan.
%logout                   {terminar esta sesión}
Host sent disconnect
```

cara para abreviar conjuntos de nombres. Se pueden utilizar los siguientes caracteres de máscara:

- ? se utiliza como comodín de caracteres individuales.
- * se utiliza como comodín para cualquier grupo de caracteres, excluyendo un punto como el primer carácter de un nombre. Esto es para impedir que archivos del sistema tales como .login o .cshrc se borren accidentalmente.
- [] (con una lista o gama de caracteres dentro de los corchetes) sirve como comodín para cualquier carácter individual con uno de los caracteres encerrados entre los corchetes, como [a,e,f], [A-Z], etc.

Para ver cómo funcionan, consideremos algunos de los comandos disponibles para tratamiento de archivos.

La instrucción ls se emplea para listar los archivos y subdirectorios de un directorio, y, al igual que muchas instrucciones Unix, ls puede tomar varias opciones (el conjunto completo se indica en el recuadro). Hay dos instrucciones que le permiten listar el contenido de un archivo: cat, que también se puede utilizar para concatenar archivos, y more, que le permite contemplar sólo una pantalla de información por vez.

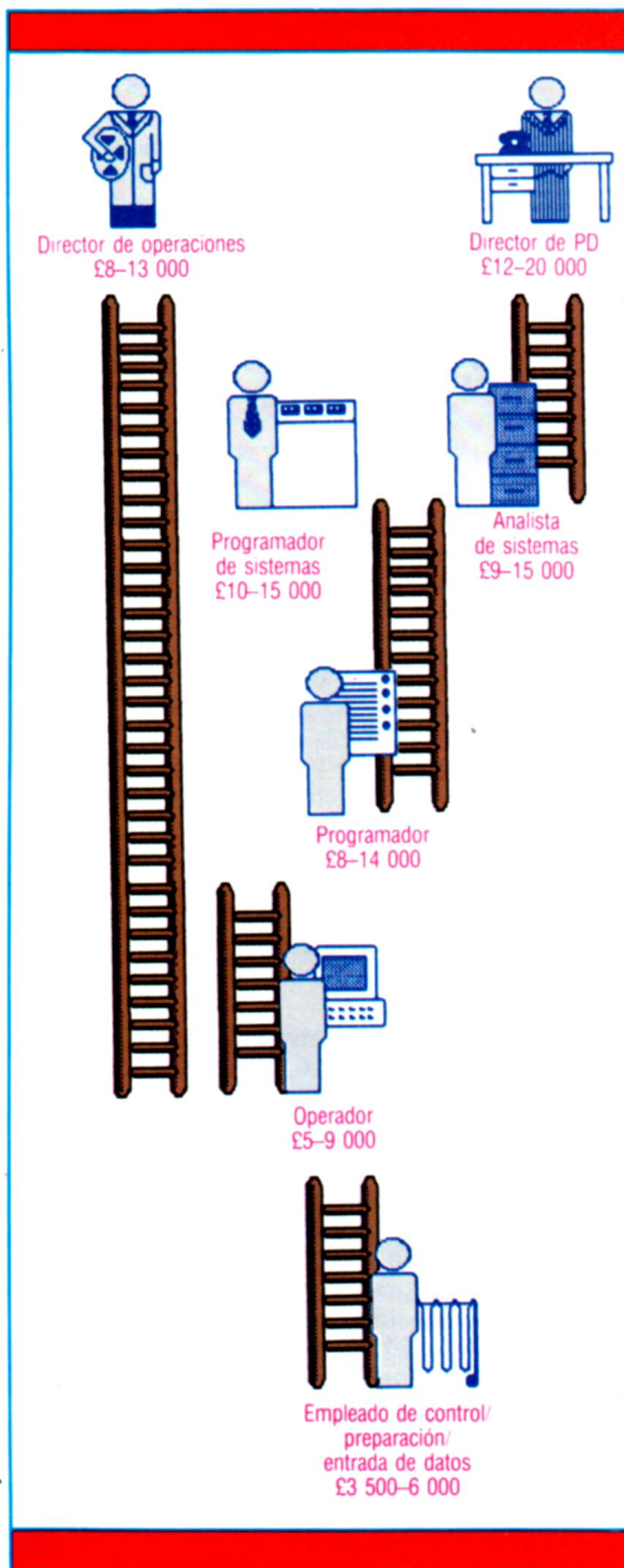
Los usuarios pueden crear y suprimir sus propios subdirectorios a su entera voluntad. La instrucción mkdir creará un nuevo subdirectorio del directorio actual, y rmdir lo eliminará, siempre que no se hayan dejado en él archivos o subdirectorios.

El directorio de trabajo actual se puede cambiar utilizando la instrucción cd. Si el nuevo archivo de trabajo es un subdirectorio del antiguo, se da el nombre del directorio; pero si el nuevo directorio está en algún otro sitio, se ha de dar un nombre de camino completo. El uso de cd sin nombre de directorio siempre le remite de nuevo al directorio base (aquel con el cual usted se conectó originalmente).

Ofrecemos una sesión con el Unix a modo de ejemplo utilizando algunos de estos comandos. Se han insertado observaciones (encerradas entre llaves {}) para explicar lo que está sucediendo; las mismas no se producen en un diálogo real.

Proceso de datos

Proceso de datos (PD) es un sector de la industria informática con una demanda aparentemente inagotable de nuevos trabajadores



Escalera al éxito
El PD ofrece numerosas oportunidades laborales, que van desde el papel ejecutivo del gerente de departamento hasta el desafío intelectual de la programación de sistemas. Nuestro diagrama refleja los distintos caminos para hacer una carrera, junto con una indicación de la escala salarial vigente en Gran Bretaña para cada empleo

Al ser Gran Bretaña uno de los países más avanzados en el campo de la informática, lo que allí suceda en este ámbito tiende a tener resonancia, en mayor o menor medida, en los restantes países europeos. Por este motivo es interesante dar una mirada a la actualidad educacional y laboral en este país en lo que respecta a la informática.

Cada semana, en las revistas *Computing* y *Computing Weekly* aparecen más de 100 páginas de anuncios de trabajo ofreciendo puestos en proceso de datos con salarios que parten de un mínimo de £10 000 a £15 000 (una libra esterlina equivale a unas 215 pts.). Veamos las oportunidades que ofrece este campo, cuáles son los caminos que conducen a tales carreras y cómo puede introducirse el novato en la industria.

Primero hemos de preguntarnos qué es realmente el proceso de datos. En esencia, el PD supone el tratamiento, recuperación y almacenamiento de información relacionada con una aplicación. Gran parte del PD gira alrededor de registros de rutina: llevar las cuentas o la nómina de una empresa, por ejemplo.

Una carrera en PD no significa que automáticamente el postulante tendrá acceso al último equipo en materia de hardware de microordenador. En la mayoría de los sitios de PD, el micro se sigue viendo como un juguete y el 90% del trabajo se lleva a cabo en miniordenadores u ordenadores centrales, suministrados por fabricantes como IBM, ICL y Burroughs.

El dibujo (izquierda) muestra las principales áreas de empleo y los "peldaños" en el sector de PD. Las mismas, no obstante, no son categorías rígidas ni expeditas. En algunos sitios, por ejemplo, una persona combinará el papel de analista y programador, en otros el de operador y programador. En general, sin embargo, éstas son las categorías laborales en las que se podría incluir el 90% de las personas de la industria.

En la parte inferior de la escalera están los empleados de *preparación de datos*, *entrada de datos* o *control de datos*. Estos trabajos implican un escaso conocimiento real, si no ninguno en absoluto, sobre informática, y tal como revelan los niveles de salarios, están en manos de empleados no cualificados. De hecho, el trabajo supone "digitar" datos en una máquina a través del teclado almacenándolos sobre un disco o cinta. También implica preparar los impresos (cortarlos, etc.) y enviarlos. En los lugares más pequeños, este aspecto del trabajo lo puede realizar el personal de secretaría con la ayuda de operadores y programadores. Es discutible si un empleado de preparación de datos se puede realmente considerar como miembro del personal de PD. En muchos lugares, la preparación de datos se considera como una labor no cualificada y sin perspectivas, que no abre las puertas a un puesto como programador u operador.

El trabajo del operador

El *operador*, la siguiente etapa hacia arriba partiendo del empleado de preparación, es responsable de operar el ordenador propiamente dicho. En consecuencia, la labor del operador se puede comparar con la del conductor de un automóvil, mientras que se puede considerar al programador como un oficial de navegación. El operador es responsable de



Analista de sistemas
Intenta aportar soluciones de PD a los problemas de los otros departamentos

Gerente de PD
Responsable de representar al departamento ante la junta, dirigir al personal, organizar la formación, etc.

Programador novato
Probablemente formado en FORTRAN o COBOL, y comprometido en desarrollar software de aplicaciones

Programador de sistemas
Trabajando ya sea en ensamblador o bien en otro lenguaje compilado de bajo nivel, quizá C o FORTH

Operador
A veces reemplazado por un programador. Responsable de instalar y operar el hardware

Empleado de entrada de datos
A menudo asistido, o incluso sustituido, por personal de secretaría. Responsable de la entrada de datos y de distribuir salidas a otros departamentos

Director de operaciones
Responsable de la operación del hardware, de organizar las existencias y el mantenimiento de los equipos. Informa al gerente de PD

Kevin Jones

Estaciones de acción
La ilustración muestra el interior de un típico departamento de PD de una gran compañía. La posición de tales departamentos se ha erosionado de manera significativa en los últimos años a medida que más jefes de departamentos han ido ganando acceso a sus propios micros de escritorio. Sin embargo, el PD sigue siendo una de las mayores fuentes de empleo y ofrece importantes oportunidades para hacer una carrera

inicializar físicamente las tareas en las máquinas, cargando cintas y discos, cargando papel en las impresoras y poniendo las máquinas en funcionamiento y desconectándolas.

Tradicionalmente, la transferencia desde el lado del operador al de la programación era relativamente sencilla, pero esta tradición está desapareciendo rápidamente a consecuencia de la "descualificación" del papel del operador y el creciente número de programadores formados y con titulación. De hecho, en el ámbito de los operadores hay muchísimas personas tratando de apartarse del campo. La facilidad con que se pueda pasar de operador al campo de la programación depende mucho de la actitud de la gerencia. Si bien no se debe descartar de buenas a primeras ser operador durante un breve período, es importante evaluar las posibilidades de promoción antes de aceptar el puesto.

La mayoría de las personas que parecen tener una carrera a largo plazo en PD intentarán ascender por la escalera hasta el siguiente peldaño: *el programador*. En comparación con la programación en un micro, la programación en un entorno PD tiende, sin embargo, a ser una actividad mucho más especializada.

Como programador novato es probable que usted aprenda COBOL o bien FORTRAN (o posiblemente PASCAL), lo que le permitiría escribir software de aplicaciones. Sin embargo, en muchos lugares, y según el hardware utilizado, hay otro estrato de programación: el del programador de sistemas. El software de sistemas se sitúa entre el sistema operativo del ordenador y el software de aplicaciones que se esté ejecutando, y casi siempre se escribe en un lenguaje de bajo nivel específico de la máquina, casi con toda seguridad ensamblador. No obstante, el C está adquiriendo creciente difusión en este campo debido a su velocidad y portabilidad.

Muchos programadores seguirán ascendiendo hasta convertirse en analistas de sistemas, el más reciente de los principales papeles laborales en PD. Esencialmente, el analista de sistemas se ocupa de decidir con los usuarios la clase de información que requieren o que podría serles útil. La tarea requiere capacidad para hablar con gerentes que quizá no posean conocimientos sobre informática, y para explicarles lo que es y no es posible. Muchas personas desconfían y temen a los ordenadores, viéndolos como una amenaza para su futuro laboral. El analista de sistemas ha de ser capaz de vencer esta desconfianza y hacerse una idea exacta de lo que supone una tarea determinada. Por este motivo, el análisis requiere gran aptitud para la comunicación (y mucha paciencia y diplomacia).

Rutas diferentes

Como programador, usted puede elegir entre dos caminos alternativos para hacer una carrera: convertirse en especialista de software o en analista de sistemas. La ruta que elija dependerá mucho de su carácter y su actitud ante la vida. Si es un solitario que no necesita mucho contacto humano en su vida laboral, entonces se sentirá feliz de permanecer en la programación. De hecho, si usted encuentra el área adecuada en la cual especializarse, la programación puede ser sumamente lucrativa y muy gratificante desde el punto de vista intelectual. Si, por el contrario, le agrada trabajar con otras personas y no se resigna a la idea de contemplar una pantalla por el resto de sus días, el análisis de sistemas podría ser la respuesta.

El gerente de PD lleva a cabo tres funciones. Con la ayuda del analista de sistemas y los programadores senior, el gerente elige y evalúa el hardware y software necesarios para la instalación. En segundo lugar, es responsable de dirigir a quienes trabajan en el departamento de PD y asegurarse de que reciban el adiestramiento adecuado. Por último, representa al departamento ante el mundo exterior, informando ya sea a la junta o al director de servicios de administración.

Hay cuatro formas básicas de convertirse en programador: usted puede progresar desde operador (no es una ruta que aconsejamos especialmente), puede introducirse en el campo a raíz de un curso de especialización, como graduado en informática, o como un graduado en prácticas con una titulación que no sea especialmente relevante (o como alguien que posea vasta experiencia comercial pero ninguna experiencia previa en informática).

Es un hecho incuestionable que la programación (y el PD en general) se está convirtiendo rápidamente en una profesión para graduados. Los graduados en informática pueden realmente seleccionar y elegir; incluso los graduados en inglés o en alguna ciencia social pueden ser capaces de obtener un adiestramiento siempre que superen la prueba fundamental. Los graduados en matemáticas o ciencias con alguna experiencia en un ordenador central de una universidad tienen ante sí un brillante comienzo.

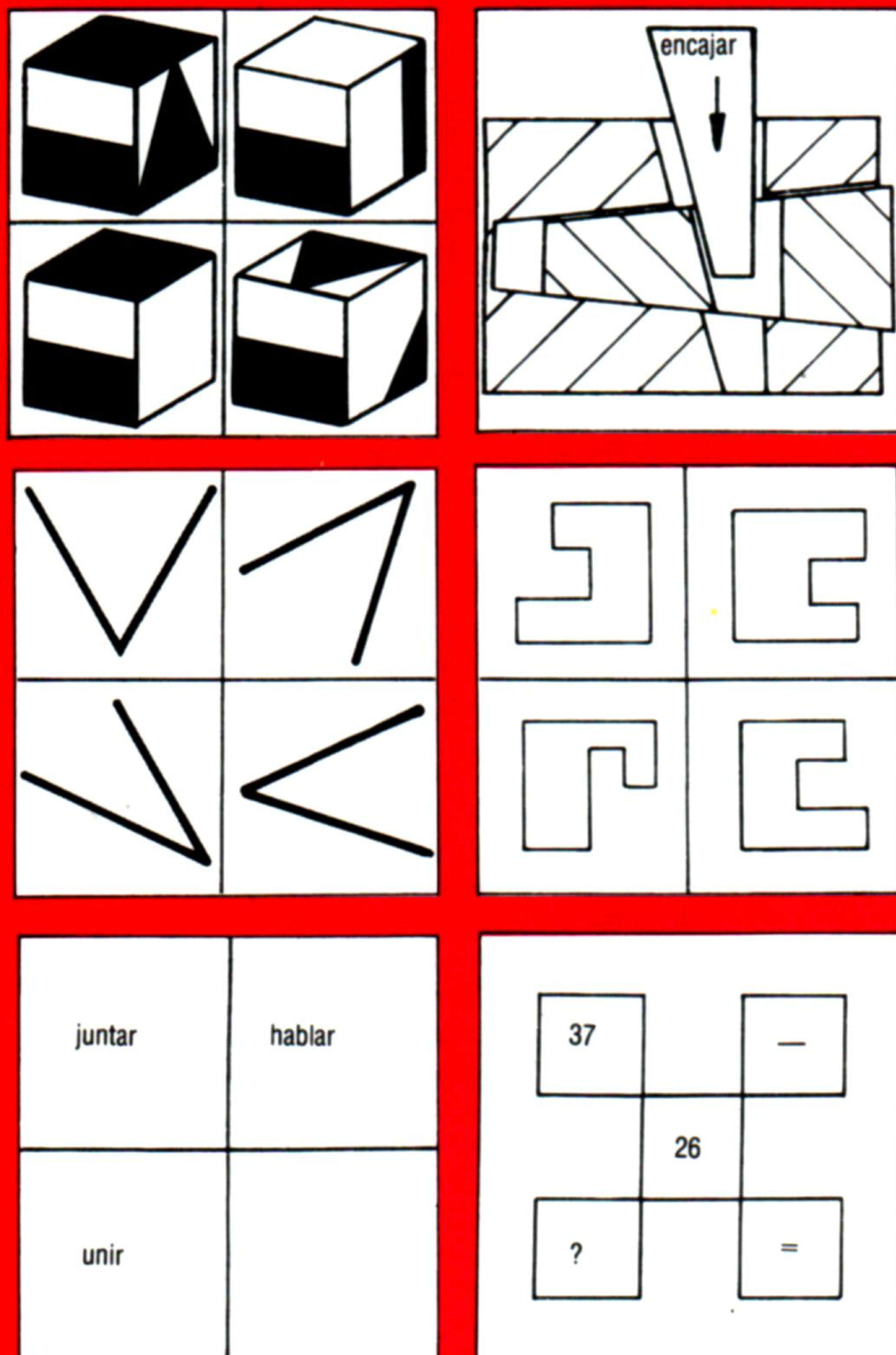
El PD no es un coto cerrado a quienes sean un poco mayores y quienes posean una experiencia útil en la industria. La firma Ford, y otras numerosas empresas automovilísticas, han adiestrado en programación a directores de líneas con un buen conocimiento de la industria del motor. Numerosas empresas agrupan a programadores experimentados con gerentes con la esperanza de obtener una mejor identificación de las necesidades del usuario.

El progreso en una carrera de PD depende en gran medida del nivel de adiestramiento que usted pueda recibir. El adiestramiento en alguna rama secreta del software de sistemas, por ejemplo, podría proporcionarle aptitudes que en Estados Unidos representan la posibilidad de acceder a un puesto con una remuneración de $4 000 o $7 000 al mes (un dólar norteamericano equivale a unas 150 pts). Pruebe de averiguar cuáles son los antecedentes de su potencial patrón respecto al adiestramiento antes de compromoterse con él. Algunas compañías le ofrecen a su personal una formación profesional mínima, o esperan que ellos aprendan en el trabajo, en la creencia generalizada de que un empleado bien adiestrado permanecerá poco tiempo. En general, cuanto mayor sea la empresa, mejor será el adiestramiento.

Aparte de la formación, el otro factor prioritario que incide en los niveles de salario es la experiencia con el ordenador. En general, es bastante difícil saltar de la máquina de un fabricante a la de otro: si usted ha estado trabajando con máquinas ICL es difícil conseguir empleo en un punto IBM. Esto es reflejo del hecho de que cada fabricante ha desarrollado software de sistemas diferentes y que éstos difieren considerablemente entre sí. Sería arriesgado obtener un empleo para trabajar con una máquina de cierta antigüedad producida por un fabricante de poco éxito, puesto que obviamente en otros lugares no habría mayor demanda para las aptitudes que usted pudiera desarrollar. Sería preferible, entonces, trabajar en un punto IBM (más del 80% de todos los ordenadores centrales del mundo son IBM).

Pero Well y Burroughs también cuentan con grandes bases de usuarios. Posiblemente la firma ICL esté en peligro de perder su dominio en el mercado británico de ordenadores centrales y no sería una buena decisión si usted tuviera intención de trabajar en el extranjero. Ha de pensárselo detenidamente antes de iniciar una carrera que suponga especialización en hardware Univac, NAS o Kienzle, por ejemplo.

Más allá del coeficiente intelectual
Con frecuencia los jefes de personal se muestran reacios a confiar exclusivamente en la entrevista con el interesado para la selección de postulantes a un empleo, y exigen que éstos realicen uno o más tests para determinar su idoneidad. Estos tests, por lo general diseñados por psicólogos profesionales, en cierto sentido son similares a los tests de coeficiente intelectual, pero van más lejos en su evaluación de aptitudes relacionadas con el puesto de trabajo. La ilustración muestra una representación simplificada de algunos de los problemas con que se puede encontrar un postulante a aprendiz técnico. De izquierda a derecha y de arriba abajo: razonamiento espacial, comprensión mecánica, estimación visual, conocimiento espacial, razonamiento verbal y cálculo numérico



Cortesía de Saville and Holdsworth, Psicólogos Profesionales, 81 High Street, Esher, Gran Bretaña

Abrir un camino

En MS-DOS se pueden organizar lógicamente software y datos colocando todo tipo de archivos en un árbol jerárquico de subdirectorios

Las utilidades del Unix pwd (*print working directory*: imprimir directorio de trabajo) y tree (visualizar la estructura del directorio) no siempre las proporcionan los OEM que suministran MS-DOS, pero hay otras alternativas disponibles. Incluyendo los caracteres $p en el comando prompt, se visualizará el nombre de camino completo (desde la raíz de la unidad actual) como parte del aviso DOS después de que termine la ejecución de cada comando.

Supongamos, no obstante, que estamos trabajando en un subdirectorio llamado, pongamos por caso, TT (de tratamiento de textos) y queremos formatear un disco y hacer una copia de un nuevo documento. Dar la instrucción format b: no sirve de nada porque la utilidad transitoria FORMAT.EXE se halla en un subdirectorio llamado sistema. Debemos decir \sistema\format b:, utilizando el nombre de camino completo. Si otro programa, llamado WCOUNT.EXE (*word count*: contar palabras) estuviera en la unidad C en un subdirectorio varios niveles más abajo, nuevamente tendríamos que dar el nombre de camino completo para que se encontrara y ejecutara. De modo que contar la cantidad de palabras de un documento llamado informe17.doc podría suponer tener que teclear:

```
C:\texto\herramientas\wcount informe17.doc
```

Afortunadamente, hay una forma muy simple de sortear este problema. El comando residente path visualizará o creará una lista de caminos por defecto en los que el MS-DOS buscará siempre que un comando no reconocido no esté situado en el directorio actual. Solo, path visualizará el mensaje No path antes de que se hayan especificado caminos por defecto. Si todos los programas y utilidades de más frecuente uso estuvieran incluidos en los directorios que acabamos de mencionar, podríamos impartir la instrucción:

```
path=A:\sistema;C:\texto\herramientas
```

Ahora, cualquiera sea el disco o directorio actual, el procesador de instrucciones encontrará cualquier programa en el directorio actual o en alguno de los otros dos especificados. La búsqueda se dirige por el mismo orden en el que se especifican los caminos; de modo que, por ejemplo, si se hubiera de reproducir el nombre de un programa, se ejecutaría la primera pareja.

Una palabra de atención: *no* se deben insertar espacios en la lista de argumentos, porque son considerados como terminadores. El comando:

```
path C:\sistema;A:\milib;C:\texto\herramientas
```

no conseguiría, por tanto, encontrar nada que no estuviera ya sea en el directorio actual o bien en la unidad C del directorio del sistema.

Las versiones de MS-DOS inspiradas en el Unix (de la versión 2.0 en adelante) poseen la capacidad de *redirigir* la E/S desde y hacia tanto dispositivos del sistema como archivos de disco. Los dispositivos estándares del sistema tales como CON (la consola) y PRN (la impresora) se manejan exactamente del mismo modo que como se trataría un archivo de texto en disco. Utilizando la instrucción residente type, normalmente se visualizaría el contenido de un archivo de texto en la pantalla, pero la salida se podría redirigir, con el símbolo *chevron* (>), a cualquier dispositivo de archivo que quisiéramos:

```
type informe17.doc>prn
```

iniciaría la impresión y produciría una salida impresa del archivo. El comando:

Más comandos residentes

He aquí otro listado de algunos de los comandos residentes del MS-DOS. Todos los comandos que vienen a continuación están incorporados en COMMAND.COM, el equivalente del MS-DOS al intérprete de línea de comando del CP/M, o CCP.

	Función
cls	Limpiar la pantalla
prompt	Cambiar el aviso del sistema
pwd	Imprimir directorio de trabajo
re (o rmdir)	Suprimir un direct. (debe estar vacío)
ver	Imprimir el núm. de versión MS-DOS
vol	Imprimir el ID de volumen

No todos los sistemas, sin embargo, tendrán todos estos comandos. Por ejemplo, ped (tomado del Unix) no es esencial, dado que se puede utilizar el comando prompt para visualizar el camino actual, como en:

```
prompt $p
```

Aludiendo la "p", en este caso, a *path* (camino). Otros caracteres que vayan tras el símbolo $ en un argumento prompt pueden tener significados especiales, y algunos de los mismos son:

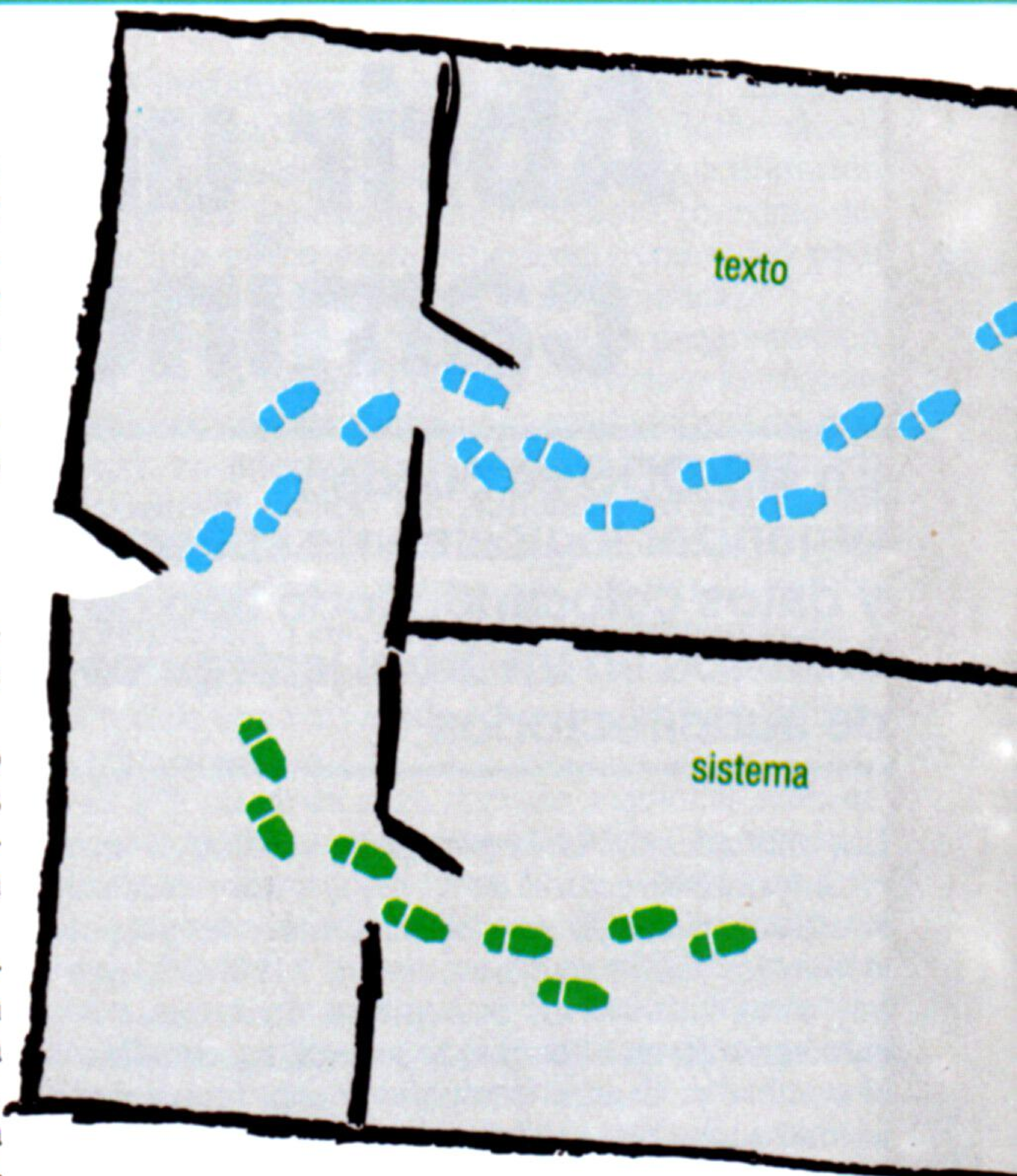
d	Imprimir la fecha
t	Imprimir la hora
_	Enviar un CR/LF para crear una nueva línea
e	Enviar un carácter escape

La secuencia de escape, tal como la utiliza un terminal ANSI estándar, se puede utilizar para cambiar los atributos de la VDU, de modo que:

```
prompt $t on $d $_$e[7m $p $e [m
```

imprimirá la hora y fecha en una línea, y el camino actual en video invertido en la siguiente. De manera que:

```
15:42:36 on 17-11-85
A:\SYSTEM\UTILS
```




```
dir a:> c:nov22.dir
```

listaría el contenido del directorio, no a la VDU como siempre, sino a un archivo de la unidad C llamado nov22.dir. Utilizando de esta manera la redirección de salida, se puede lograr que cualquier programa o utilidad envíe su salida a cualquier archivo o dispositivo del sistema.

El doble *chevron*, >>, produce la misma redirección de salida pero sin sobreescribir el contenido previo de un archivo. De modo que después de:

```
dir b:>> c:nov22.dir
```

el archivo nov22.dir contendrá una lista de los archivos y subdirectorios de los directorios actuales de las unidades A y B.

Utilizado de forma aislada esto es apenas sólo conveniente, pero en combinación con unas pocas "herramientas de software" el empleo de la redirección de E/S puede casi transformar el MS-DOS en un eficaz lenguaje de programación interactivo.

Algunas de las herramientas de software más eficaces son programas simples que introducen algún cambio en los datos leídos a partir de la entrada estándar antes de pasárselos a la salida estándar. Éstos se denominan *filtros*, por la evidente analogía con la electrónica. Es muy fácil escribir filtros, incluso para los programadores recién iniciados, con la salvedad de que han de estar escritos en un lenguaje compilado de modo que el código máquina producido sea "independiente" y se pueda ejecutar tal como lo hace cualquier utilidad del sistema sin que esté presente un intérprete o el cód. fuente.

El Unix y las versiones 2/3 de MS-DOS poseen la capacidad de conectar archivos con cualquiera de las rutinas de manipulación de caracteres para los dispositivos de entrada *y* salida estándares del sistema. Para redirigir la entrada se utiliza <. En este caso, se le puede indicar a todo programa que espera datos de entrada desde el teclado que los obtenga de un archivo. Esto incluye instrucciones tanto transitorias como residentes. Por ejemplo:

```
fecha<fecha.hoy
```

aceptará una representación en serie de una fecha de un archivo llamado fecha.hoy, apareciendo en la pantalla, como siempre, el mensaje de salida normal producido por la instrucción. Esto puede ser muy útil si sólo se dispone de un reloj de software, que se restablece cada vez que se vuelve a cargar el sistema. La inclusión de la línea anterior en el archivo autoexec.bat evita el tener que volver a digitar continuamente la fecha.

Hay algunos filtros útiles que se suministran como utilidades transitorias del MS-DOS. Uno de ellos (MORE.COM) toma cualquier texto entrado y simplemente lo copia en la salida estándar. No obstante, cada 23 líneas MORE visualiza el mensaje:

```
- -More- -
```

y aguarda a que se pulse una tecla, permitiendo de este modo contemplar una pantalla de texto cada vez al ritmo deseado. Por ejemplo:

```
more<informe.doc
```

Naturalmente, la redirección de salida no es especialmente útil con MORE.

Una de las utilidades transitorias del MS-DOS más útiles es SORT.EXE. Éste es un filtro que lee líneas de texto y, antes de copiarlas sin modificación en la salida estándar, las clasifica por orden alfabético. Si deseáramos un listado de directorio clasificado de todos los archivos de texto, por ejemplo, se podría conseguir con:

```
dir*.txt> temp.dir
sort < temp.dir
del temp.dir
```

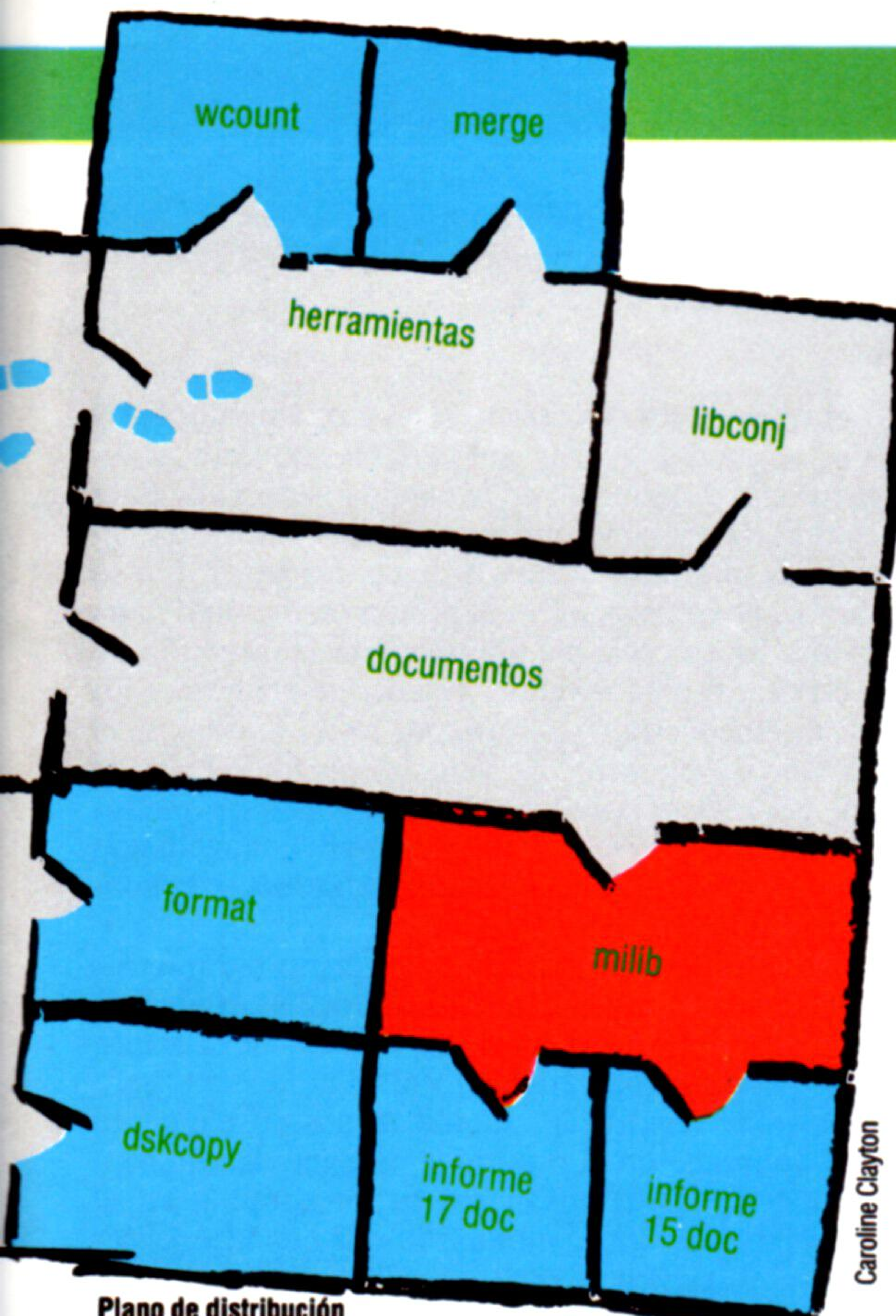
El MS-DOS, al igual que el Unix, permite llevar a cabo estos procesos en una operación mediante un *tubo* o *tubería*. Se utiliza el símbolo | para indicar que la salida de un proceso se debe redirigir a un canal o corriente que ha de actuar como la fuente de entrada para un ulterior proceso. En el caso que

Planta de filtración

Un "filtro" simple puede ser un programa que simplemente tome una entrada estándar y altere, añada o suprima ciertos caracteres antes de pasar el resultado (filtrado) a la salida estándar. Consideremos un filtro simple (llamado pipz) para impedir que lleguen a la salida caracteres con códigos ASCII de 128 en adelante. Los archivos de texto producidos con el *WordStar* y con ciertos otros programas a menudo contienen tales caracteres, puesto que el bit más significativo se puede establecer con fines de formateado. Se puede realizar fácilmente una copia "en limpio" de un archivo de texto que originalmente contuviera tales códigos, mediante una instrucción para redirigir los canales tanto de entrada como de salida:

```
pipz<archmalo.txt>archbueno.txt
```

Este sencillo programa (en versión en PASCAL) "pliega" todos los valores de caracteres a la gama de códigos ASCII normal (hasta 127) y, por lo tanto, simula el empleo de la opción [z] con la utilidad pip del CP/M, poniendo a cero el bit *high* de



Plano de distribución
La estructura de archivos jerárquica del MS-DOS se puede considerar como una serie de habitaciones conectadas entre sí. La entrada a ciertas habitaciones sólo se puede efectuar a través de otras. Las habitaciones "muertas" representan archivos reales de programas, en vez de directorios (en color azul). Supongamos que estamos trabajando dentro del subdirectorio llamado "milib" (en color rojo) en un archivo de texto llamado "informe17.doc" Los programas de utilidades tales como contadores de palabras (wcount) y los formateadores de disco residen en otras "habitaciones". Para no tener que especificar el camino completo desde la raíz cada vez que queramos utilizar una utilidad, el MS-DOS proporciona un comando path que define caminos por defecto. Para disfrutar del acceso directo a los directorios de utilidades podríamos impartir el comando: path\sistema;\ texto\ herramientas, que especificaría dos caminos alternativos para la utilidad requerida

cada byte de carácter. Esto es particularmente útil porque la instrucción copy del MS-DOS no dispone de esta opción. El archivo se podría listar, asimismo, directamente a la impresora mediante:

```
pipz<archmalo.txt>prn
```

Un ejemplo de filtro en PASCAL:

```
PROGRAM PipZ (input,output);
{plegar cars a gama ASCII 0..127}
VAR
  c:char;
BEGIN
  WHILE NOT EoF (input) DO
    BEGIN
      WHILE NOT EoLn (input) DO
        BEGIN
          read (input,c);
          write (output,
            chr (ord(c) MOD 128))
        END;

      ReadLn (input);
      WriteLn (output)
    END
END.
```

acabamos de ilustrar, por lo tanto, sólo necesitamos decir:

```
dir*.txt|sort
```

El MS-DOS manipula de forma invisible todo archivo temporal que se requiera, y lo redirige automáticamente cuando termina una tubería.

La instrucción sort puede tomar opciones, incluyendo /r (para clasificar por orden inverso) y /+n (donde n es cualquier número), indicando la posición de carácter de cada línea en la cual comenzarían las comparaciones para clasificación. Los detalles del directorio del MS-DOS incluyen los tamaños de los archivos expresados en bytes (empezando por la posición del 15.º carácter) y también la hora y fecha de creación. Podríamos, en consecuencia, producir un listado en salida impresa del directorio clasificado con los archivos más grandes listados primero:

```
dir|sort/r/+ 15 >prn
```

Otro potente filtro del DOS (FIND.EXE) se puede utilizar, por ejemplo, para hallar series en cualquier archivo de texto dado, simplemente mediante:

```
find"sort "C:\artículos\MSDOS4.txt
```

En la salida se lista cada línea del archivo que contenga la serie "sort " (una sola vez, incluso cuando en la misma línea apareciera en varias ocasiones). Observe que el espacio que sigue a "sort significa que no se hallarán sorted ni sorting. Ni tampoco se localizaría Sort, porque find es sensible a las mayúsculas o minúsculas.

Esta deficiencia se puede superar fácilmente escribiendo nuestro propio filtro simple (LOWER.EXE) para convertir todas las entradas a minúsculas. Entubándola "en serie" antes de otras instrucciones entubadas se suprime la sensibilidad a los tipos de letra de todas las otras herramientas:

```
lower<meeting.doc|find"metal"|more
```

listaría todas las líneas del documento (meeting.doc) que hicieran referencia a metal, metálico o metalurgia (con o sin mayúsculas) y haría una pausa tras visualizar cada 23 de tales líneas.

La potencia de las tuberías y la redirección queda bien ilustrada con instrucciones tales como:

```
dir*.pas|find"–08–85"|sort>ptn
```

que imprime todos los archivos fuente en PASCAL (.pas) creados en agosto por orden alfabético. La salida de find se podría listar con los números de línea antepuestos (la opción /n) o se podría suprimir la visualización de líneas y obtener con /c una cuenta total del número de ocurrencias. La opción /v halla todas las líneas que *no* contengan la serie.

Supongamos que tenemos un archivo de texto (compañía.dat) que contiene algunos nombres y otros detalles, incluyendo salarios, que comienza en la columna 25:

```
lower<compañía.dat|find"smith"/v|sort/+25|more
```

produciría una lista de todos los que *no* se llamaran Smith o Smithers, etc., clasificada por orden descendente de salarios.

Mediante el empleo de tubos y redirección hemos creado un nuevo programa en la línea de comando construyéndolo en el acto a partir de simples herramientas de software de filtro.

Punteros

Nos corresponde examinar el procedimiento de declaración y el empleo de los punteros

Diseñado como sustituto del lenguaje ensamblador, el C posee una gama de facilidades inexistentes en otros lenguajes de alto nivel. Una de éstas es la facilidad para aludir directamente a direcciones máquina. El PASCAL posee tipos *puntero*, pero la diferencia entre éstos y las direcciones máquina no está clara en absoluto. En C, el tipo puntero alude a direcciones reales, si bien hay que recordar que en una máquina multitarea, particularmente aquella que utilice memoria virtual, el verdadero conocimiento de la dirección numérica de una variable normalmente no es de especial utilidad.

No obstante, el C posee la capacidad de aludir a direcciones absolutas y, por tanto, se puede utilizar para activar dispositivos de hardware directamente. Toda variable, del tipo que sea, tiene una dirección que se puede asignar a un puntero. A menudo es más conveniente, en especial con las series (matrices de caracteres), emplear aritmética de puntero para acceder a elementos de la serie en lugar de utilizar los habituales subíndices de matriz.

El carácter * se utiliza para declarar una variable puntero, de modo que:

```
int*p;
```

declararía una variable puntero p. Observe la declaración int: los punteros sólo pueden apuntar a un tipo de variable determinado (el tipo "básico"), de modo que si quisiéramos que un puntero accediera a variables de tipo char, deberíamos entrar:

```
char*p;
```

y así sucesivamente. El carácter *, cuando es utilizado de este modo, significa simplemente "crear un puntero", de modo que en esta etapa el puntero propiamente dicho no apunta hacia ningún lugar.

Para inicializar un puntero en un valor requerido, usamos el carácter &, el cual, colocado delante de una variable, tiene el efecto de devolver la dirección de ésta en vez de su valor. De modo que:

```
p=&i;
```

establecería p como un puntero de i. En este punto, usted debe observar que el carácter * también se puede utilizar para indicar el valor al que apunta una variable puntero.

En este caso:

```
i = *p;
```

almacenaría el *valor* apuntado por p en i, mientras que:

```
i = p;
```

almacenaría la *dirección* apuntada por p. Usted, por supuesto, necesitará un *cast* para evitar mensajes de error del compilador al manipular direcciones absolutas, como en:

```
p = (int*) 1000;
```

Los operadores de incremento y decremento ++ y −− se pueden utilizar con variables puntero, y aumentan o disminuyen en la cantidad adecuada para el tipo de variable que se esté apuntando. En un sistema que utilice enteros de cuatro bytes, por lo tanto, si p es un puntero al primer elemento de una matriz de enteros, p++ aumentará la dirección de p en cuatro para apuntar al siguiente elemento. Lo mismo sucedería si utilizáramos p=p+1: sumaría el tamaño del elemento de dato adecuado en lugar de apenas 1. Esto pone de relieve el hecho de que los punteros no son lo mismo que enteros, aun cuando los valores en ellos almacenados sean verdaderamente números enteros.

Los punteros se pueden pasar como parámetros en llamadas a funciones. Ello ofrece la facilidad de pasar por referencia, es decir, el valor del puntero se le pasará a una variable local de la función, pero este valor seguirá apuntando al mismo elemento que apuntaba en el programa de llamada. En consecuencia, los cambios que se introduzcan en el valor almacenado continuarán allí cuando se devuelva el control.

Existe una correspondencia muy íntima entre punteros y matrices. Cuando se declara una matriz, el nombre de la matriz sin subindexar es, en realidad, un puntero al primer elemento de la matriz. Dicho en otras palabras, dado int a [100], *p, luego p=a; y p=&a[0]; son totalmente equivalentes. Esto significa que con frecuencia el acceso a elementos de la matriz (o de submatrices) se puede realizar más eficazmente mediante aritmética de punteros que mediante subíndices.

Utilización de series

Las series son, simplemente, matrices unidimensionales de variables de tipo char. Dado que su uso es fundamental para muchas aplicaciones, el C, no obstante, proporciona unas pocas facilidades especiales y funciones de biblioteca para llevar a cabo los tipos normales de proceso de series. En C, una serie se compone de una matriz de char en la que los caracteres reales que componen la serie van seguidos inmediatamente por el carácter nulo \0 (ASCII cero). Esto podría ocurrir en cualquier punto de la matriz, de modo que tenemos al menos la ilusión de series dinámicas de longitudes variables aun cuando las matrices subyacentes hayan de ser de longitud fija. Recuerde que el carácter \0 ocupará una posición de carácter en la matriz, de modo que ésta debe ser definida con al menos un elemento más que la cantidad máxima de caracteres que vaya a albergar.

Se pueden utilizar constantes en serie, si están encerradas entre comillas dobles, y se las puede asignar a una matriz de char de longitud adecuada; el terminador \0 se añadirá automáticamente. La variable a la cual se asignan puede ser ya sea una matriz o bien un puntero a char:

```
char *s;
*s="abc";
```

hace que los cuatro caracteres a, b, c y \0 se alma-

Funciones de manipulación de series

strcmp (s1,s2)—s1 y s2 son punteros a char (series), se devuelve un valor entero que es menor que cero si s1<s2, cero si s1=s2 y mayor que cero si s1>s2
strncmp(s1,s2,n)—Similar a strcmp excepto que, como máximo, se comparan n caracteres
strlen(s)—Donde s es un puntero a char, devuelve la longitud (en enteros) de la serie
index(s,c)—Donde s es un puntero a char y c es un char, devuelve un puntero a la primera aparición de c en s, o NULL si no aparece ninguna vez
rindex(s,c)—Similar a index, excepto que la búsqueda se lleva a cabo desde el carácter situado más a la derecha
strcat(s1,s2)—Añade una copia de s2 al final de s1, devolviendo s1. Se da por sentado que s1 es suficientemente larga como para contener todos los caracteres
strncat(s1,s2)—Añade como máximo n caracteres no nulos al final de s1, devolviendo s1
strcpy(s1,s2)—Copia el contenido de s2, hasta la '\', en s1. Se da por sentado que s1 es suficientemente larga. El cont. previo de s1 se pierde. Se devuelve el valor de s1.
strncpy(s1,s2,n)—Copia como máximo n caracteres de s2 en s1, añadiendo una '\' a menos que n> = la longitud de s2



Burbujeante

```
burbuja(a,n)
int a [],ponerenorden();
/* a es una matriz de n enteros a clasificar*/
{
int i,j;
   for(i=0;i<n – 1;++i)
      for (j=n – 1;i<j; ––j)
         ponerenorden(&a[j – 1],&a[j]);
/*paso de las direcciones de los elementos
de la matriz como parámetros*/
}
ponerenorden(x,y)
int*x,*y,swap();
/*x e y son punteros a enteros*/
{
      if(*x>*y)
         swap(x,y);
}
swap(x,y)
int*x,*y;
{
      int temp;
      temp= *p;
      *p=*q;
      *q=temp;
/*observe que las direcciones de los dos enteros se
pasan por valor pero la función puede aludir a los
valores reales almacenados allí aun cuando estén
estrictamente fuera del ámbito*/
}
```

Apuntando por números
Nuestro listado muestra la estrecha relación existente entre los punteros y las matrices en c, implementando el tradicional algoritmo de clasificación por el método de la burbuja utilizando una matriz de enteros. Con frecuencia los punteros del c pueden ser más eficaces que los subíndices para acceder a los elementos de las matrices

En tres actos

Aquí vemos tres etapas en el uso de un puntero teórico CharPtr (utilizado para hacer referencia a distintos elementos de la matriz en serie STRING []). Observe que tras la inicialización, CharPtr devuelve una dirección, mientras que *CharPtr devuelve el valor retenido en esa dirección. Los punteros se pueden incrementar y reducir para apuntar a distintos elementos de una matriz: el valor del puntero se ajustará de acuerdo al "tipo de dato básico" dado en la sentencia de declaración. Por ejemplo, un tipo básico de char dará por resultado ajustes de pasos de un solo byte, e int (en la mayoría de los sistemas) producirá ajustes de dos bytes

DECLARACIÓN

Char	*CharPtr;

TIPO BÁSICO: cada puntero necesita una declaración de tipo que indique el tipo de datos al cual apunta

En una declaración, *indica que la variable es un puntero

CharPtr

Cuando se declara por primera vez, el puntero no apunta a ningún lugar. Sólo entra en acción tras habérsele asignado un valor

FE00	FE01	FE02	FE03	FE04	FE05	FE06	FE07	FE08
A	Espacio	S	T	R	I	N	G	0

INICIALIZACIÓN

CharPtr = &String[0]

El símbolo &, cuando va delante de una variable, devuelve la DIRECCIÓN de ésta, en vez de su valor

CharPtr

Ahora CharPtr retiene la DIRECCIÓN del elemento cero de la matriz STRING []

FE00	FE01	FE02	FE03	FE04	FE05	FE06	FE07	FE08
A	Espacio	S	T	R	I	N	G	0

VALOR

Avariable = *CharPtr

El símbolo * hace que el puntero devuelva el CONTENIDO de la posición a la cual apunta, en vez de la dirección

CharPtr

Anteponiéndole un * a CharPtr podemos acceder al valor retenido en la dirección apuntada

FE00	FE01	FE02	FE03	FE04	FE05	FE06	FE07	FE08
A	Espacio	S	T	R	I	N	G	0

cenen en cuatro posiciones consecutivas empezando en la dirección actual de s, que parte apuntando al carácter a. Después de ++s, s apuntaría entonces al b, y así sucesivamente.

Los punteros son variables como cualquier otra, de modo que ocupan un espacio de almacenamiento que tiene una dirección. En consecuencia, es bastante posible tener punteros a punteros, y así sucesivamente. Esto puede dificultar bastante las cosas, pero son muy pocos los usos de matrices de punteros. Uno de ellos, en particular, es en el reconocimiento de argumentos de línea de comando, es decir, caracteres que se digitan en la misma línea que la llamada al programa.

Hay dos parámetros especiales que se le pueden pasar a la función principal, que proporciona acceso a los mismos. Se trata de argc, que da una cuenta de la cantidad de argumentos de línea de comando (estando los argumentos separados por al menos un espacio), y argv, que es una matriz de punteros a series con elementos argc. Cada elemento de argv apuntará al primer carácter en esa serie de números de la línea de comando. El siguiente y breve programa ilustra esto simplemente imprimiendo los argumentos de línea de comando, uno en cada línea:

```
main(argc,argv)
int argc;
char * argv[];
/* observe que esta declaración da una matriz
de punteros*/
{
   int i;
   for (i=1;i< argc;i++)
      printf("%s\n",argv[i]);
/* observe que el formato '%s' requiere un puntero a
una serie*/
}
```

Caer en una subrutina

Llegados a este punto, examinaremos la instrucción JSR ("Jump to subroutine": salto a la subrutina) y su utilidad práctica

Para mostrar cómo actúa esta instrucción, comencemos con un sencillo programa que transforma una tabla de 10 palabras de longitud mediante la fórmula:

```
TARRAY[I]:=ARRAY[I]^2+20*ARRAY[I]
```

y obtiene la suma de todos los elementos que componen TARRAY colocándola en la posición SUM. La fórmula parece un tanto complicada, pero lo único que expresa es "toma cada elemento de la tabla, elévalo al cuadrado, suma 20 veces dicho elemento, y después almacena el resultado en la casilla correspondiente de la nueva tabla, TARRAY".

La implementación del 68000 para esta transformación se muestra en el "listado uno". Para comprobar la corrección del programa debemos examinar los valores de TARRAY para los datos de comprobación (del 1 al 10) almacenados en ARRAY. Si usted ejecuta el programa, los valores almacenados serán, naturalmente, 21, 44, 69, 96, etc. Observe que el problema no especifica el intervalo de números que se han de usar en ARRAY. Si nuestras respuestas han de ser correctas, el resultado de la transformación (y la suma) se ajustarán a la longitud de una palabra, es decir, 16 bits.

Pero ésta no es, ni mucho menos, la única solución del problema, pues existen otros muchos caminos para llegar al mismo resultado. Esto ya depende del diseño de programa, y es en esta fase del proceso de codificación cuando cobra importancia el empleo de la subrutina, especialmente en el lenguaje assembly.

Hay muchos métodos para diseñar programas, pero una manera de alcanzar lo que pudiéramos llamar un programa "aseado" (al tiempo que resulta fácilmente comprensible cuando se vuelve a revisar) es el llamado *desglose funcional*. Se trata de un método que tiene por finalidad la elaboración de secciones de programas bien definidas que realicen ciertas funciones con los datos. Las funciones se describen mediante verbos tales como "calcular", "trasladar" o "comprobar", y estas secciones de código, en un lenguaje de alto nivel, se denominan *procedimientos*, con los datos facilitados a través de los *parámetros*.

Al nivel de la codificación en ensamblador, la única construcción modular adecuada disponible es la subrutina, que en el 68000 se llama por medio de:

```
JSR     SUBR
```

Esto altera el flujo del programa para realizar un salto a la sección de código etiquetada como SUBR. El código que está en SUBR se ejecutará hasta encontrar RTS, en cuyo punto el control del flujo del programa se devuelve a la instrucción siguiente a JSR (el mnemónico JSR significa *Jump to SubRoutine* [salto a la subrutina] y RTS quiere decir *ReTurn from Subroutine* [retorno de subrutina]).

Volvamos a nuestro programa ejemplo. Supongamos que en vez de realizar el cálculo de elevar al cuadrado, multiplicar y sumar "en una línea", deseamos "descomponer" esto con una llamada a la subrutina para conseguir un mejor diseño. Acabaremos con un programa algo así como el mostrado en el "listado dos". Todo lo que ha cambiado ha sido que en la línea 6 tenemos una llamada de subrutina a CALC, y lo que antes aparecía en las líneas de la 6 a la 9 se contiene ahora en las líneas 12 a la 15, con una RTS colocada al final.

La ventaja está en que ahora poseemos una sección de código bien definida (las líneas de la 12 a la 16), que realiza una operación específica sobre los datos pasados en D1 (un parámetro). No sólo es un buen diseño sino que podemos llamar a este código, CALC, ¡tantas veces como precisemos para que repita el mismo cálculo sobre otros datos siempre contenidos en D1 (y depositados en D2)!

Otro detalle a tener en cuenta cuando se implementan subrutinas, en especial en entornos de multiusuario, es que se tiene por una buena práctica el disponer de una sola entrada y una sola salida para las subrutinas. Esto quiere decir que debemos evitar las llamadas a otras subrutinas desde dentro de una subrutina. Tampoco es recomendable implementar un gran número de *returns* condicionados, que hagan la depuración una tarea difícil.

Ventajas de las subrutinas

La subrutina no sólo favorece el buen diseño de un programa, sino que proporciona programas económicos en lo que a ocupación de memoria se refiere. Esto se debe a que ahora no será necesario repetir el código de CALC cada vez que debamos emplearlo. Es adaptable tanto en el sentido de poderla llamar desde cualquier punto del programa como el que si deseamos introducir alguna modificación sólo tendremos que mirar un fragmento determinado del programa para realizarla.

Por ejemplo, supongamos que deseamos esta transformación:

```
D2:=D1^2-20*D1
```

la nueva subrutina CALC cambiará su línea 15 así:

```
SUB     D1,D2
```

y asunto concluido. Es más, el programa, además de ser adaptable, resulta fácil de depurar. Por

Listado uno

```
* Transformación de ARRAY en TARRAY mediante
* TARRAY[I]:=ARRAY[I]^2+20*ARRAY[I]
* Ambas tablas constan de 10 elementos
* Los registros empleados son:
*   A0 apunta a ARRAY
*   A1 apunta a TARRAY
*   D0 es un contador de bucle
*   D1 es un almacenamiento temporal
*   D2 es una copia de ARRAY[I]

           ORG     $1000
1 START    MOVEQ   #10,D0        * establece contador bucle
2          LEA     ARRAY,A0      * establece el primer puntero
3          LEA     TARRAY,A1     * y el segundo
4          CLR     SUM           * prueba si la suma es cero
5 LOOP     MOVE    (A0)+,D1      * toma ARRAY[I]
6          MOVE    D1,D2         * y la copia
7          MULS    D2,D2         * hace el cuadrado de ARRAY[I]
8          MULS    #20,D1        * D1 llega a ser 20 veces ARRAY[I]
9          ADD     D1,D2         * suma elementos
10         MOVE    D2,(A1)+      * almacena en TARRAY[I]
11         ADD     D2,SUM        * y guarda el total dinámico
12         SUBQ    #1,D0         * decrementa contador bucle
13         BNE     LOOP          * itera si no es el fin
14         TRAP    #0            * salida a monitor
15 ARRAY   DC.W    1,2,3,4,5,6,7,8,9,10
16 TARRAY  DS.W    10
17 SUM     DS.W    1
18         END
```

Notas al programa:
Entre las líneas 1 y 4 se inicializan las variables empleadas por el programa; después, en la línea 5, cada elemento de ARRAY es cargado en D1 mediante el direccionamiento indirecto con posdecremento. En la línea 6 se hace una copia en D2, de manera que no tenemos que acceder a la tabla de nuevo. Las líneas 7 y 8 forman dos componentes de la transformación en D1 y D2, mediante MULS, y la línea 9 suma ambas. La línea 10 almacena el resultado en el elemento correspondiente de TARRAY, y la línea 11 suma el resultado en la suma dinámica, SUM. Finalmente, las líneas 12 y 13 son sentencias de control del bucle que permiten transformar sólo los 10 elementos (entre 5 y 12/13 se establece, en efecto, un bucle FOR con el contador inicializado en la línea 1)

Listado dos

```
           ORG     $1000
1 START    MOVEQ   #10,D0
           LEA     ARRAY,A0
           LEA     TARRAY,A1
           CLR     SUM
5 LOOP     MOVE    (A0)+,D1
6          JSR     CALC          * llamada a subrut. para calcular
7          MOVE    D2,(A1)+      * nuevo valor de la tabla
8          ADD     D2,SUM
9          SUBQ    #1,D0
10         BNE     LOOP
11         TRAP    #0
12 CALC    MOVE    D1,D2         * subrut. para calc. D1^2+20*D1
13         MULS    D2,D2         * y poner el valor en D2
14         MULS    #20,D1
15         ADD     D1,D2
16         RTS                   *retorno al programa que llamó
```

ejemplo, podemos probar la validez de CALC mediante todo tipo de valores que daremos a D1 y examinar los resultados en D2 sin mayores complicaciones con otros trozos del programa o con datos irrelevantes. Sólo debe haber una entrada a la subrutina (en la etiqueta de dirección de la subrutina) y sólo una salida (en la instrucción RTS). Así podemos afirmar que la estructuración de nuestros programas con subrutinas facilitan las pruebas y la depuración.

Parece, pues, que así se satisfacen los criterios de un "buen programa", por lo menos desde el punto de vista de la economía, adaptabilidad y fiabilidad. Pero el empleo de subrutinas comporta también ciertas desventajas. Para percatarnos de ellas (y examinar la transmisión de datos por medio de parámetros) es necesario echar un vistazo al mecanismo de la pila en el 68000.

El mecanismo de la pila

Cuando ejecutamos la instrucción JSR es preciso recordar la dirección de la instrucción siguiente a aquella que llama a la rutina (el enlace con la subrutina). La instrucción JSR hace que se coloque en la pila la dirección de esa instrucción siguiente (se necesitan cuatro bytes para la dirección de una palabra larga completa) y que el contador del programa (PC) se cargue con la dirección de la subrutina. Cuando se ha ejecutado RTS en la subrutina, el PC se cargará con dicha dirección que será sacada de la pila, y la ejecución continuará después de la llamada a la subrutina. Toda esta manipulación de la dirección se hace al margen del usuario, pues el 68000 se encarga de todo. Pero hay efectos colaterales que el programador debe conocer.

Ante todo, hay que cerciorarse de que el puntero de la pila, SP (o bien, A7 en el 68000), es activado correctamente, de lo contrario podemos sobreescribir el código o los datos, ¡o incluso violar las direcciones del hardware! El modo habitual de hacer esto es:

```
STACK   EQU   $1000       *pone la pila a 1000 hexa
BEGIN   LEA   STACK,SP    *inicializa el puntero
                           de la pila
```

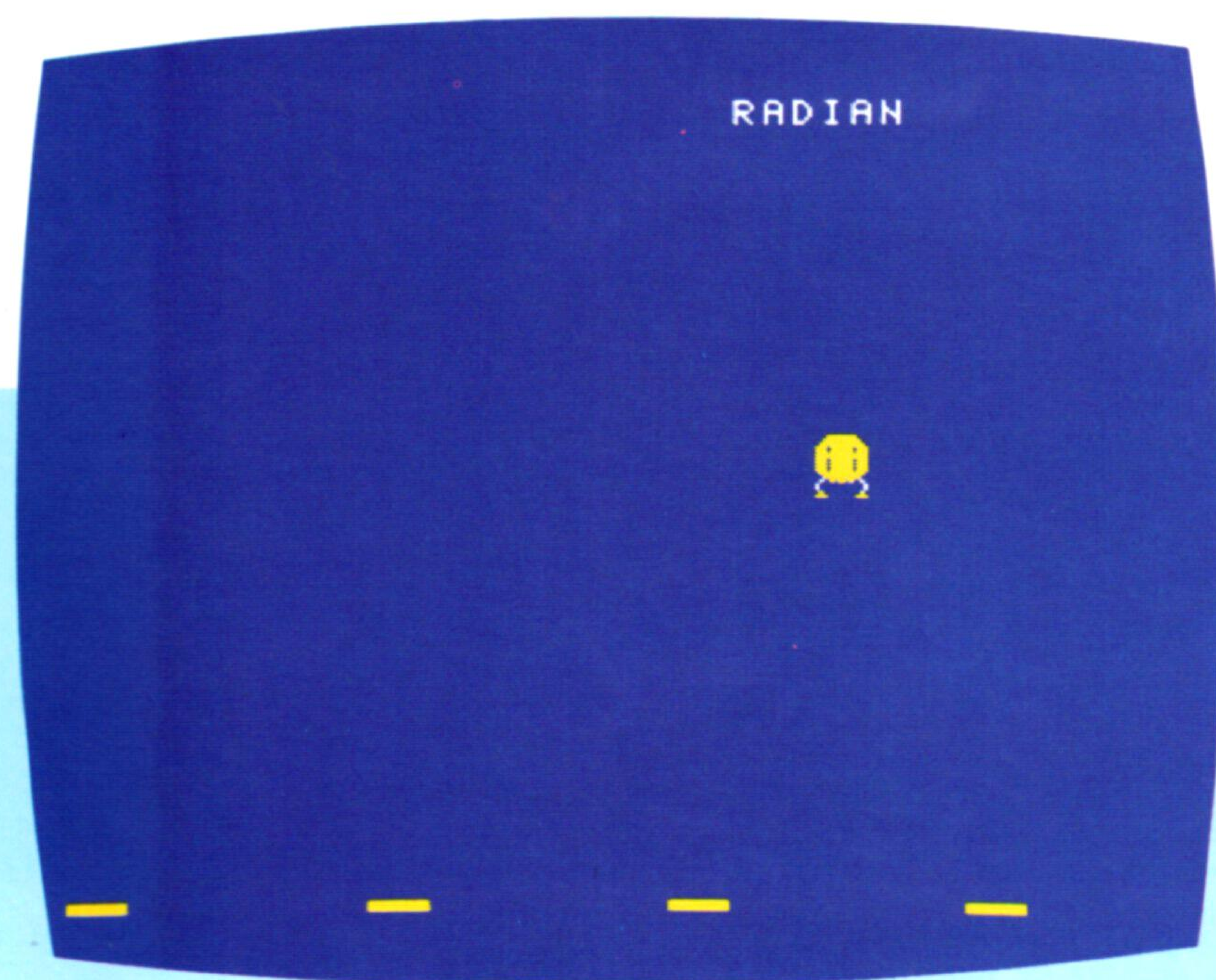
y la pila irá pasando del 1000 al cero (porque cada *push* o envío a la pila es un predecremento).

En segundo lugar, ¡debemos estar seguros de que la pila no haya crecido demasiado! Debemos calcular la cantidad de pila que necesitamos usar, lo cual es difícil. Para programas relativamente sencillos con sencillas llamadas a subrutinas y empleos de la pila, es posible tener una respuesta exacta mediante el examen del código. Para programas *anidados* (es decir, con subrutinas que llaman a otras subrutinas) o *recursivos* (una subrutina se llama a sí misma), el problema de dar un tamaño a la pila puede ser casi irresoluble, y se han de emplear técnicas de ensayo y error.

En el próximo capítulo daremos ejemplos pormenorizados de llamadas a subrutinas y, en particular, nos ceñiremos a los métodos para pasar los datos a y desde la subrutina, empleando en ciertos casos algunas instrucciones únicas del 68000.

Aterrizaje

He aquí un juego de acción que está siempre de actualidad. Esta versión, escrita por Pierre Monsaut, está destinada al microordenador EXL 100

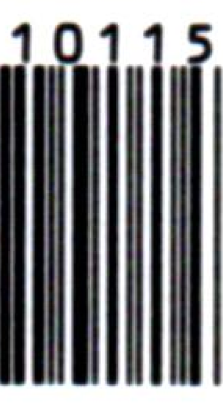


Después de un largo viaje sin gravedad, no resulta nada fácil aterrizar suavemente con una nave espacial; pero gracias a su ordenador está en condiciones de efectuar un entrenamiento sin riesgos. Debe posar su nave sobre una de las cuatro zonas destinadas al efecto. Usted puede dirigirse a la derecha y a la izquierda con ayuda de las teclas de control del cursor.

```
100 REM **************
110 REM * ATERRIZAJE *
120 REM **************
130 GOSUB 590
140 FOR I=1 TO 100
150 NEXT I
160 IF DL<0 THEN DL=0
170 GOSUB 690
180 FOR Q=2 TO 20
190 FOR I=0 TO DL*10
200 NEXT I
210 CALL KEY1(D3,D4)
220 NH=NX
230 NX=NX+(D3=131)-(D3=129)
240 IF NX<2 TTHEN NX=2
250 IF NX>38 THEN NX=38
260 LOCATE (Q-1,NH)
270 PRINT C$;
280 LOCATE (Q,NH)
290 PRINT C$;
300 LOCATE (Q,NX)
310 PRINT N$;
320 LOCATE (Q+1,NX)
330 PRINT M$;
340 NEXT Q
350 IF INT ( (NX-3)/10)=(NX-3)/10 THEN DL=DL-1:GOTO 140
360 CALL COLOR("1Yb")
370 LOCATE (Q,NX)
380 PRINT C$;
390 LOCATE (Q-1,NX)
400 PRINT C$;
410 LOCATE (Q+1,NX-1)
420 PRINT H$;
430 CALL COLOR("1RG")
440 LOCATE (7,6)
450 PRINT "SU NAVE SE HA ESTRELLADO";
460 LOCATE (12,12)
470 PRINT "SCORE :";S-1;
480 LOCATE (15,11)
490 PRINT "OTRA ?";
500 FOR I=1 TO 100
510 NEXT I
520 CALL KEY1(D3,D4)
530 IF D3<>255 THEN 520
540 CALL KEY1(D3,D4)
550 IF D3=255 THEN 540
560 IF D3<>78 THEN RUN
570 CLS
580 END
590 CLS ("Ybb")
600 GOSUB 800
610 B$=CHR$(32)
620 N$=CHR$(100)&CHR$(101)
630 C$=B$&B$
640 M$=CHR$(102)&CHR$(103)
650 A$=CHR$(104)&CHR$(104)
660 H$=CHR$(103)&CHR$(101)&CHR$(100)&CHR$(102)
670 S=0
680 RETURN
690 CALL COLOR("1Yb")
700 FOR I=1 TO 20
710 NEXT I
720 CLS
730 FOR I=0 TO 3
740 LOCATE (22,3+I*10)
750 PRINT A$;
760 NEXT I
770 S=S+1
780 NX=INTRND(35)+2
790 RETURN
800 CALL CHAR(100,"0000001F3F7FEFEFFFEF")
810 CALL CHAR(101,"000000E0F0F8DCDCFCDC")
820 CALL CHAR(102,"EFEFEF7F3F4BB08040E0")
830 CALL CHAR(103,"DCDCDCF8F0480404081C")
840 CALL CHAR(104,"FFFFFF00000000000000")
850 DL=20
860 RETURN
```


mi COMPUTER
Editorial Delta, S.A.




10115
9 788485 822836